Documentação do Trident 26.10

Trident

NetApp
November 10, 2026

This PDF was generated from https://docs.netapp.com/pt-br/trident/index.html on November 10, 2026.
Always check docs.netapp.com for the latest.

Índice

Documentação do Trident 26.10 . . . 1
Notas de lançamento . . . 2
O que há de novo . . . 2
O que há de novo na versão 26.10.1 . . . 2
O que há de novo em 26.10 . . . 2
O que há de novo na versão 26.06.2 . . . 6
O que há de novo na versão 26.06.1 . . . 7
O que há de novo em 26.06 . . . 7
O que há de novo na versão 26.02.1 . . . 10
O que há de novo em 26.02 . . . 12
O que há de novo em 25.10 . . . 14
Mudanças em 25.06.2 . . . 16
Mudanças em 25.06.1 . . . 16
Mudanças em 25.06 . . . 17
Mudanças em 25.02.1 . . . 19
Mudanças em 25,02 . . . 19
Mudanças em 24.10.1 . . . 21
Mudanças em 24,10 . . . 21
Mudanças em 24,06 . . . 23
Mudanças em 24,02 . . . 24
Mudanças em 23,10 . . . 25
Mudanças em 23.07.1 . . . 25
Mudanças em 23,07 . . . 25
Mudanças em 23,04 . . . 26
Mudanças em 23.01.1 . . . 27
Mudanças em 23,01 . . . 28
Mudanças em 22,10 . . . 29
Mudanças em 22,07 . . . 30
Mudanças em 22,04 . . . 31
Mudanças em 22.01.1 . . . 31
Mudanças em 22.01.0 . . . 32
Mudanças em 21.10.1 . . . 32
Mudanças em 21.10.0 . . . 33
Problemas conhecidos . . . 33
Encontre mais informações . . . 34
Versões anteriores da documentação . . . 35
NetApp Trident suporte para sistemas de storage ONTAP ASA r2 . . . 35
Operações suportadas . . . 36
Operações não suportadas . . . 36
Problemas conhecidos . . . 36
Os volumes permanecem anexados após a atualização do Trident 22.10 ou anterior . . . 36
VolumeSnapshots não atingindo o estado ReadyToUse . . . 36
Restaurar backups Restic de arquivos grandes pode falhar . . . 37

Comece agora . . . 38
Saiba mais sobre o Trident . . . 38
Saiba mais sobre o Trident . . . 38
Arquitetura da Trident . . . 39
Conceitos . . . 42
Início rápido para Trident . . . 46
O que se segue? . . . 47
Requisitos da Trident . . . 47
Informações críticas sobre o Trident . . . 47
Frontens suportados (orquestradores) . . . 47
Backends suportados (armazenamento) . . . 48
Suporte ao Trident para virtualização KubeVirt e OpenShift . . . 48
Suporte do Trident para Kubernetes em Kubernetes e OpenShift planos de controle hospedados . . . 49
Requisitos de recursos . . . 50
Sistemas operacionais de host testados . . . 50
Configuração de host . . . 51
Configuração do sistema de storage . . . 51
Portas Trident . . . 51
Imagens de contêineres e versões correspondentes do Kubernetes . . . 51
Instale o Trident . . . 53
Saiba mais sobre a instalação do Trident . . . 53
Informações críticas sobre Trident 26.10 . . . 53
Antes de começar . . . 53
Escolha o método de instalação . . . 53
Escolha o modo de instalação . . . 55
Selecione o processo com base no seu método e modo . . . 56
Movendo-se entre os métodos de instalação . . . 56
Outras opções de configuração conhecidas . . . 56
Instale usando o operador Trident . . . 57
Implantar manualmente o operador Trident (modo padrão) . . . 57
Implantar manualmente o operador Trident (modo off-line) . . . 61
Implantar operador Trident usando Helm (modo padrão) . . . 67
Implantar operador Trident usando Helm (modo off-line) . . . 76
Personalizar a instalação do operador Trident . . . 85
Instale usando o tridentctl . . . 96
Instale o Trident usando o tridentctl . . . 96
Personalize a instalação do Trident tridentctl . . . 100
Instale usando o operador certificado OpenShift . . . 101
Instale o Trident usando o OpenShift OperatorHub . . . 101
Mude para o operador Trident certificado OpenShift . . . 104
Use o Trident . . . 108
Prepare o nó de trabalho do Trident . . . 108
Selecionar as ferramentas certas . . . 108
Detecção de serviço de nós . . . 108
Volumes NFS . . . 109

Volumes iSCSI . . . 109
Volumes NVMe/TCP . . . 113
SCSI em volumes FC . . . 114
Prepare-se para provisionar volumes SMB . . . 116
Provisionar storage de objetos com COSI . . . 117
Visão geral . . . 117
Requisitos e limitações . . . 118
Habilitar COSI . . . 118
Configure um armazenamento de objetos do StorageGRID . . . 119
Configurar um armazenamento de objetos ONTAP S3 . . . 119
Crie uma BucketClass e solicite um bucket . . . 120
Conceder acesso de carga de trabalho a um bucket . . . 121
Configurar e gerenciar backends . . . 122
Configure os backends do Trident . . . 122
Azure NetApp Files . . . 122
Google Cloud NetApp volumes . . . 139
Configurar um back-end NetApp HCI ou SolidFire . . . 177
Controladores SAN ONTAP . . . 182
Drivers nas ONTAP . . . 211
Utilize o posicionamento balanceado com os backends ONTAP . . . 245
Amazon FSX para NetApp ONTAP . . . 248
Crie backends com kubectl . . . 281
Gerenciar backends . . . 287
Criar e gerenciar classes de armazenamento . . . 297
Crie uma classe de armazenamento do Trident . . . 297
Gerencie as classes de armazenamento do Trident . . . 299
Provisionar e gerenciar volumes . . . 301
Provisione um volume do Trident . . . 301
Mover um volume . . . 304
Expandir volumes do Trident . . . 307
Entenda os limites do subsistema RWX NVMe . . . 317
Escalabilidade do controlador . . . 318
Expansão de volume automática . . . 323
Configurar o dimensionamento automático de volume do ONTAP . . . 330
Gerenciar políticas de crescimento automático . . . 334
Importar volumes do Trident . . . 343
Personalize os nomes e rótulos de volume do Trident . . . 354
Compartilhe um volume NFS do Trident entre namespaces . . . 357
Clone volumes do Trident em namespaces diferentes . . . 361
Replique volumes usando o SnapMirror . . . 363
Use a topologia CSI . . . 370
Trabalhe com snapshots do Trident . . . 377
Trabalhe com snapshots de grupo de volume do Trident . . . 384
Gerenciar e monitorar o Trident . . . 389
Atualize o Trident . . . 389

Atualize o Trident . . . 389
Atualize com o operador . . . 390
Atualize o Trident com tridentctl . . . 395
Gerenciar o Trident usando o tridentctl . . . 395
Comandos e sinalizadores globais . . . 395
Opções de comando e sinalizadores . . . 397
Suporte ao plugin . . . 402
Monitorar as operações de storage do Trident . . . 402
Visão geral . . . 402
Passo 1: Defina um alvo Prometheus . . . 402
Passo 2: Crie um Prometheus ServiceMonitor . . . 402
Passo 3: Consultar métricas do Trident com PromQL . . . 404
Saiba mais sobre telemetria Trident AutoSupport . . . 405
Desativar métricas do Trident . . . 406
Desinstale o Trident . . . 406
Determine o método de instalação original . . . 407
Desinstale a instalação de um operador Trident . . . 407
Desinstale uma `tridentctl` instalação . . . 408
Trident para Docker . . . 409
Pré-requisitos para implantação . . . 409
Verifique os requisitos . . . 409
Ferramentas NVMe . . . 411
Ferramentas FC . . . 412
Implante o Trident . . . 413
Método de plug-in gerenciado Docker (versão 1,13/17,03 e posterior) . . . 413
Método tradicional (versão 1,12 ou anterior) . . . 415
Atualize ou desinstale o Trident . . . 418
Atualização . . . 418
Desinstalar . . . 419
Trabalhe com volumes . . . 420
Crie um volume . . . 420
Remova um volume . . . 421
Clonar um volume . . . 421
Acesse volumes criados externamente . . . 422
Opções de volume específicas do condutor . . . 422
Recolher registos . . . 427
Recolha registos para resolução de problemas . . . 427
Dicas gerais de solução de problemas . . . 428
Gerenciar várias instâncias do Trident . . . 428
Etapas para o plugin gerenciado do Docker (versão 1,13/17,03 ou posterior) . . . 428
Passos para o tradicional (versão 1,12 ou anterior) . . . 429
Opções de configuração de armazenamento . . . 429
Opções de configuração global . . . 429
Configuração ONTAP . . . 430
Configuração do software Element . . . 438

Problemas e limitações conhecidos . . . 439
A atualização do plug-in de volume do Docker do Trident para 20,10 e posterior a partir de versões mais antigas resulta em falha de atualização com o erro de nenhum arquivo ou diretório. . . . 439
Os nomes dos volumes devem ter um mínimo de 2 carateres. . . . 440
O Docker Swarm tem certos comportamentos que impedem o Trident de suportá-lo com cada combinação de armazenamento e driver. . . . 440
Se um FlexGroup estiver sendo provisionado, o ONTAP não provisiona um segundo FlexGroup se o segundo FlexGroup tiver um ou mais agregados em comum com o FlexGroup sendo provisionado. . . . 440
Práticas recomendadas e recomendações . . . 441
Recomendações de implantação do Trident . . . 441
Implante em um namespace dedicado . . . 441
Use cotas e limites de intervalo para controlar o consumo de armazenamento . . . 441
Melhores práticas para a configuração de storage do Trident . . . 441
Visão geral da plataforma . . . 441
Práticas recomendadas de ONTAP e Cloud Volumes ONTAP . . . 441
Práticas recomendadas da SolidFire . . . 446
Onde encontrar mais informações? . . . 448
Integre o Trident com o Kubernetes . . . 448
Seleção e implantação do driver . . . 448
Design da classe de armazenamento . . . 451
Design de pool virtual . . . 452
Operações de volume . . . 453
Serviço de métricas . . . 457
A proteção de dados e a recuperação de desastres do Trident . . . 458
Replicação e recuperação do Trident . . . 458
Replicação e recuperação da SVM . . . 459
Replicação de volume e recuperação . . . 460
Proteção de dados do Snapshot . . . 460
Automatizando o failover de aplicações com estado usando o Trident. . . . 460
Detalhes sobre Force Detach . . . 460
Detalhes sobre o failover automático . . . 461
Segurança . . . 466
Recomendações de segurança do Trident . . . 466
Configuração unificada de chave do Trident Linux (LUKS) . . . 467
Corrija um Lease travado para volumes RWX do Linux Unified Key Setup (LUKS) . . . 474
A criptografia Kerberos em trânsito do Trident . . . 476
Proteja seus aplicativos com o Trident Protect. . . . 484
Saiba mais sobre o Trident Protect. . . . 484
O que se segue? . . . 484
Instale o Trident Protect . . . 484
Requisitos do Trident Protect . . . 484
Instale e configure o Trident Protect. . . . 488
Instale o plugin Trident Protect CLI . . . 491
Personalize a instalação do Trident Protect . . . 494
Gerenciar Trident Protect . . . 499

Gerencie a autorização e o controle de acesso do Trident Protect. 499
Monitorar recursos do Trident Protect 505
Gere um pacote de suporte Trident Protect 509
Aprimore o Trident Protect 511
Gerenciar e proteger aplicativos 513
Use objetos do Trident Protect AppVault para gerenciar buckets. 513
Criptografe os metadados do AppVault para snapshots e backups do Trident Protect 524
Defina uma aplicação para gestão com o Trident Protect. 531
Proteja aplicativos usando o Trident Protect. 536
Use snapshots de grupo de volume com Trident Protect 546
Restaurar aplicativos 551
Replique aplicações com o NetApp SnapMirror 573
Migre aplicativos usando o Trident Protect. 589
Proteja máquinas virtuais KubeVirt com Trident Protect 593
Gerenciar os ganchos de execução do Trident Protect 605
Desinstale o Trident Protect. 612
Blogs Trident e Trident Protect 614
Trident blogs 614
Blogs da Trident Protect 614
Conhecimento e apoio 616
Perguntas frequentes 616
Questões gerais 616
Instalar e usar o Trident em um cluster do Kubernetes 616
Solução de problemas e suporte 617
Atualize o Trident 618
Gerenciar backends e volumes 619
Solução de problemas 623
Resolução de problemas gerais 623
Implantação sem êxito do Trident usando o operador 625
Implantação sem êxito do Trident usando `tridentctl` 626
Remova completamente Trident e CRDs 627
Falha de desinstalação do nó NVMe com namespaces de bloco bruto RWX do Kubernetes 1,26 627
Os clientes NFSv4.2 relatam "argumento inválido" após atualizar o ONTAP quando esperavam que "v4.2-xattrs" estivesse habilitado 628
Suporte 628
Ciclo de vida do suporte da Trident 628
Auto-suporte 629
Apoio comunitário 629
Suporte técnico da NetApp 629
Para mais informações 629
Referência 630
Portas Trident 630
Visão geral 630
API REST do Trident 632
Quando usar a API REST 632

Usando a API REST . . . 632
Opções de linha de comando . . . 633
A registar . . . 633
Kubernetes . . . 633
Docker . . . 634
DESCANSO . . . 634
Objetos Kubernetes e Trident . . . 634
Como os objetos interagem uns com os outros? . . . 634
Objetos do Kubernetes `PersistentVolumeClaim` . . . 635
Objetos do Kubernetes `PersistentVolume` . . . 637
Objetos do Kubernetes `StorageClass` . . . 637
Objetos do Kubernetes `VolumeSnapshotClass` . . . 641
Objetos do Kubernetes `VolumeSnapshot` . . . 641
Objetos do Kubernetes `VolumeSnapshotContent` . . . 642
Objetos do Kubernetes `VolumeGroupSnapshotClass` . . . 642
Objetos do Kubernetes `VolumeGroupSnapshot` . . . 642
Objetos do Kubernetes `VolumeGroupSnapshotContent` . . . 643
Objetos do Kubernetes `CustomResourceDefinition` . . . 643
ObjetosTrident `StorageClass` . . . 644
Objetos de back-end do Trident . . . 644
ObjetosTrident `StoragePool` . . . 644
ObjetosTrident `Volume` . . . 644
ObjetosTrident `Snapshot` . . . 646
Objeto Trident `ResourceQuota` . . . 647
Padrões de segurança do pod (PSS) e restrições de contexto de segurança (SCC) . . . 648
Contexto de segurança do Kubernetes necessário e campos relacionados . . . 648
Padrões de segurança do pod (PSS) . . . 649
Políticas de segurança do pod (PSP) . . . 649
Restrições de contexto de segurança (SCC) . . . 651
Avisos legais . . . 653
Direitos de autor . . . 653
Marcas comerciais . . . 653
Patentes . . . 653
Política de privacidade . . . 653
Código aberto . . . 653

Documentação do Trident 26.10

Notas de lançamento

O que há de novo

As notas de versão fornecem informações sobre novos recursos, aprimoramentos e correções de bugs na versão mais recente do NetApp Trident.

> O `tridentctl` binário para Linux que é fornecido no arquivo zip do instalador é a versão testada e suportada. Esteja ciente de que o `macos` binário fornecido na `/extras` parte do arquivo zip não é testado ou suportado.

O que há de novo na versão 26.10.1

Saiba o que há de novo no NetApp Trident e Trident Protect, incluindo melhorias, correções e descontinuações.

Trident

Correções

- **Kubernetes**:
  - Corrigidos os pods de nó do Trident que forçavam a desanexação de volumes em uso na inicialização após a atualização do Trident 25.10 ou anterior, o que fazia os pods falharem com `volume … is not staged`.

O que há de novo em 26.10

Saiba o que há de novo no NetApp Trident e Trident Protect, incluindo melhorias, correções e descontinuações.

> Não atualize para o Trident 26.10.0. Durante a inicialização, os pods de nó do Trident podem desanexar volumes que ainda estão em uso. Os pods afetados podem falhar com o erro `volume … is not staged`. Esse problema foi corrigido no Trident 26.10.1. Em vez disso, atualize para o Trident 26.10.1 ou posterior.

Trident

Melhorias

- **Kubernetes**:
  - Adicionado suporte ao provisionamento dinâmico baseado em COSI de buckets de storage de objetos do StorageGRID. Para mais informações, consulte "Provisionar storage de objetos com COSI".
  - Agora há suporte à atualização periódica de LIFs de dados NVMe para detectar e se adaptar a alterações de configuração. Para mais informações, consulte "Atualizar LIFs de dados NVMe".
  - Adicionado suporte ao dimensionamento automático do ONTAP em novos volumes provisionados pelos drivers ONTAP-SAN e ONTAP-SAN-Economy. Para mais informações, consulte "Configurar o dimensionamento automático de volume do ONTAP".
  - Agora há suporte ao Kubernetes 1.37. Para versões compatíveis e imagens necessárias, consulte

"Requisitos da Trident".

  - Adicionado o reconhecimento de PVC de clone de backup para backups do CommVault e do Rubrik para ignorar a fila de recuperação de volume do ONTAP (problema "#1182").
  - Adicionada a `maxNamespacesPerSuperSubsystem` opção de backend para novos volumes NVMe RWX. Para mais informações, consulte "Limites do subsistema NVMe RWX".
  - Adicionado o registro de nós CSI baseado em CRD por meio de `TridentNode` especificação e status, substituindo o canal de retorno do controlador HTTPS para implantações do Kubernetes.
  - Adicionamos suporte ao instalador para projeção de tokens de conta de serviço com um público-alvo configurável para o Google Cloud Workload Identity Federation (WIF) no OpenShift. Para obter detalhes sobre a instalação e a configuração do backend, consulte "Configure o Google Cloud NetApp Volumes com a federação de identidade da carga de trabalho".
  - Adicionamos um `backend_state` rótulo à `trident_backend_info` métrica para identificar a integridade individual de cada backend no Prometheus (Problema "#1088"). Para obter informações sobre as métricas do Trident, consulte "Monitore o Trident".

- Adicionado suporte à criptografia LUKS para volumes RWX provisionados pelo driver ONTAP-SAN via iSCSI. Para mais informações, consulte "Usar a configuração de chave unificada do Linux (LUKS)".
- Adicionado suporte de simultaneidade GA para o driver ONTAP-NAS-FlexGroup e o Docker; a simultaneidade do ASA r2 agora está em GA, tendo saído da pré-visualização técnica. Para drivers e comportamento compatíveis, consulte "Escalabilidade do controlador".
- Adicionado suporte à expansão de volume CSI quando nem todos os caminhos iSCSI estão disponíveis, usando remoção de caminho e autorrecuperação de iSCSI. Para mais informações, consulte "Expanda volumes".
- Adicionada a anotação PVC `trident.netapp.io/unixGroupID` para definir o ID do grupo UNIX do volume do ONTAP na criação para volumes ONTAP-NAS e ONTAP-NAS-FlexGroup com root squash (Problema "#1079").
- Adicionado suporte ao Talos OS para NVMe/TCP, NFSv4 e iSCSI (Problema "#806"). Para plataformas suportadas e requisitos de nós de trabalho, consulte "Requisitos da Trident" e "Prepare o nó de trabalho".
- Adicionado o parâmetro opcional `storageAPITimeout` de backend para configurar os tempos limite dos clientes REST e ZAPI do ONTAP por backend.
- Adicionada a `useRESTSQLBypass` opção de backend do ONTAP para adicionar `SQL=false` às pesquisas de nomes REST do ONTAP.
- A importação de qtree do ONTAP-NAS-Economy foi promovida de Technical Preview para GA. Para obter instruções de importação, consulte "Importar volumes do Trident".

Correções

- **Kubernetes**:
  - Corrigido o problema em que as regras da política de exportação do ONTAP-NAS eram excluídas incorretamente durante atualizações de backend ou reinicializações do controlador quando `enableConcurrency` está `true` (problemas "#1166", "#1179"). Para a configuração da política de exportação, consulte "Exemplos e opções de configuração do ONTAP nas".
  - Corrigido um problema em que atualizações não essenciais do backend, como alterações de credencial ou de QoS, acionavam verificações de órfãos por volume que podiam bloquear operações de CSI em backends grandes. As verificações agora são executadas somente quando o prefixo da SVM ou de armazenamento é alterado.
  - Credenciais exibidas em logs de depuração foram corrigidas, incluindo dumps de solicitações gRPC

do CSI, saída do ambiente de inicialização e especificações registradas `TridentBackendConfig`. A saída de inicialização agora registra valores apenas para as variáveis do Trident `TRIDENT_*` e lista as outras variáveis por nome.

- Corrigida uma condição de corrida entre a publicação do CSI e a movimentação de volume que podia anexar um volume com informações de acesso desatualizadas. Para obter detalhes sobre a movimentação de volume, consulte "Mover um volume".
- Corrigido `NodeGetInfo` o retorno de rótulos de topologia vazios antes da conclusão do registro do nó, o que poderia causar falhas no provisionamento com reconhecimento de topologia (Problema "#1131"). Para provisionamento com reconhecimento de topologia, consulte "Provisionar armazenamento usando a topologia CSI".
- Corrigida uma rara condição de corrida de desconexão de NVMe que podia derrubar um subsistema compartilhado enquanto outro pod estava sendo conectado.
- Corrigidas falhas na publicação de clones de PVC NVMe e o mapeamento incorreto do namespace pai ao clonar um volume montado sob concorrência.
- Corrigido o problema de falhas na criação de snapshots para PVCs em backends ONTAP-SAN-Economy configurados com `nameTemplate`. Para exemplos de configuração de backend, consulte "Exemplos e opções de configuração de SAN ONTAP".
- Corrigido o problema em que os backends REST do ONTAP deixavam um FlexVol recém-criado órfão caso ele fosse excluído antes que o índice de nomes REST fosse atualizado; o Trident agora exclui volumes por UUID.
- Corrigido o problema de travamento na inicialização do controlador quando um LIF de backend está inacessível. O backend afetado é marcado `Failed` e a inicialização continua.
- Corrigido o problema de reinicialização dos pods do controlador e do nó gerenciados pelo operador a cada reconciliação quando configurados com um seletor de nó de várias chaves. Para obter informações sobre a configuração do seletor de nó, consulte "Personalizar a instalação do operador Trident".
- Corrigidos os recursos do Azure NetApp Files `TridentBackendConfig` criados pelo TridentConfigurator sem o rótulo `trident.netapp.io/configurator` exigido pela validação pós-criação.
- Corrigida a falha no registro de nós quando o serviço Trident CSI usa um ClusterIP IPv6 (problema "#1164").
- Corrigido um problema que fazia com que um nó do Trident fosse removido permanentemente quando um nó de trabalho era excluído e reintegrado enquanto publicações de volume obsoletas ainda estavam sendo despublicadas.
- Corrigida a perda de acesso de exportação NFS em volumes ONTAP-NAS em nós já anexados quando a política de exportação em cache do Trident estava desatualizada. Para configurar a política de exportação, consulte "Exemplos e opções de configuração do ONTAP nas".
- Corrigida `tridentctl` a limpeza da instalação que excluía o CRD do modelo de correção do nó, o que poderia impedir instalações posteriores com a desconexão forçada ativada. Para obter detalhes da instalação, consulte "Instale o Trident usando o tridentctl".
- Corrigimos as chamadas de exclusão de snapshots que sobrecarregavam a API do ONTAP ao adicionar um atraso de repetição. Para operações de snapshot, consulte "Gerenciar instantâneos".
- Corrigimos o comentário do auxiliar de imagem AutoSupport do gráfico Helm do operador Trident e as aspas do valor cloudProvider.
- Corrigido o tratamento de criação de volume ZAPI do ONTAP para trabalhos ocupados ou existentes, para que as novas tentativas do CSI não destruam prematuramente FlexVols nem causem loops de

criação/destruição.

  - Foi corrigido um panic ao tentar novamente snapshots de grupo de volume para um grupo de consistência que não existe mais. Para snapshots de grupo de volume, consulte "Criar instantâneos de grupos de volumes".
  - Removemos o aviso de suporte do Trident com base nas versões mínimas e máximas do Kubernetes compatíveis.

- Corrigimos uma condição de corrida em bloqueios nomeados compartilhados que podia permitir operações sobrepostas no mesmo recurso.
- Corrigida a falha na inicialização do driver ASA quando não foi possível encontrar um snapshot.
- Correção no provisionamento do GCNV em pools regionais do Flex Unified quando os nós do cluster não estão na zona de HA primária; o Trident não trata mais essa zona de HA como uma topologia CSI obrigatória. Os pools zonais permanecem inalterados. Para a configuração do Flex Unified, consulte "Configurar armazenamento em bloco (iSCSI)".
- Corrigido o problema em que volumes GCNV Flex Unified NAS e SAN eram marcados incorretamente como órfãos durante a reconciliação do backend quando a identidade canônica do backend diferia do nome do volume do Trident (Problema "#1189"). Para a configuração do backend GCNV, consulte "Configurar Google Cloud NetApp Volumes".
- Correção na detecção de NVMe/TCP em hosts onde o transporte está integrado ao kernel (Problema "#1019"). Para requisitos de nó, consulte "Prepare o nó de trabalho".
- Corrigido o problema de tentativas indefinidas de expansão do ONTAP-NAS e do ONTAP-NAS-FlexGroup quando o volume do ONTAP já era grande o suficiente para a solicitação, incluindo a reserva de snapshot. Para expansão de volume, consulte "Expanda volumes".
- Corrigimos a falha na montagem de volumes fixos devido à ausência de caminhos de junção. Para a configuração NAS do ONTAP, consulte "Configurar os drivers NAS do ONTAP".
- Corrigido o problema de falha na criação de volumes ONTAP-SAN-Economy quando o nome do `storagePrefix`volume contém hífens. Para opções de configuração do backend, consulte "Exemplos e opções de configuração de SAN ONTAP".
- Correção da publicação de clones ONTAP-NAS somente leitura com políticas de exportação automáticas, para preservar as regras de exportação do FlexVol de origem. Para a configuração do ONTAP NAS, consulte "Exemplos e opções de configuração do ONTAP nas".
- Corrigido o comportamento de nova tentativa do transporte HTTP (Problema "#1139").
- Correção na detecção de dispositivos fantasma que ignorava a remoção de dispositivos SCSI (Problema "#1161").
- Corrigido o provisionamento com simultaneidade habilitada que selecionava uma zona de topologia diferente da zona preferida do nó selecionado. Para comportamento de simultaneidade e topologia, consulte "Escalabilidade do controlador".
- Corrigido o posicionamento desigual de volumes em backends equivalentes quando a concorrência está ativada; os backends agora são selecionados aleatoriamente de forma uniforme, como no provisionamento não simultâneo. Para obter detalhes, consulte "Escalabilidade do controlador".
- Corrigido o problema de nós se registrarem com endereços IP ausentes, o que poderia remover regras da política de exportação NFS quando a tabela de endereços de rede do host mudava durante a inicialização do pod de nó do Trident.
- Corrigido o problema de criação de clones que continuava após a consulta REST e vinculava o PVC como um FlexClone não dividido; o Trident agora observa falhas posteriores no trabalho de criação.
- Corrigido o relato falso de falhas de redimensionamento no nó durante a expansão de volume do ONTAP-SAN. Para expansão de volume, consulte "Expanda volumes".

Melhorias experimentais

Não deve ser usado em ambientes de produção.

- **[Prévia Técnica] Kubernetes**: adicionado suporte ao provisionamento dinâmico baseado em COSI de buckets de storage de objetos do ONTAP S3. Para mais informações, consulte "Provisionar storage de objetos com COSI".

Trident Protect

Melhorias

- Adicionado suporte para criptografar metadados de snapshot e backup do AppVault. Use uma senha de cofre personalizada para cada AppVault. A rotação da senha do cofre criptografa novamente a chave de criptografia de metadados e atualiza todos os repositórios Kopia e Restic no AppVault. Para saber mais, consulte "Criptografe os metadados do AppVault para snapshots e backups do Trident Protect".
- Adicionamos suporte para reter o endereço MAC das VMs do KubeVirt durante a restauração, o que é útil quando uma licença de aplicativo está vinculada a um endereço MAC. Para habilitar isso, defina a anotação protect.trident.netapp.io/retain-mac-address como "true". Para saber mais, consulte "Utilize as configurações avançadas de restauração do Trident Protect.".
- Adicionamos suporte a snapshots de grupo de volume, que capturam as persistent volume claims relacionadas de uma aplicação no mesmo ponto no tempo. Defina grupos de volume em `spec.volumeGroups` no recurso personalizado Application por rótulo, por máquina virtual KubeVirt ou ambos. Para saber mais, consulte "Use snapshots de grupo de volume com Trident Protect".
- Adicionamos suporte para failover para um snapshot anterior de um aplicativo replicado, permitindo a recuperação a partir de um ponto no tempo anterior à ocorrência de danos ou comprometimento dos dados. Para saber mais, consulte "Faça failover para um snapshot anterior usando o Trident Protect".

Correções

- Corrigidas as instalações do Trident Protect que falhavam no `trident-protect-pre-upgrade-hook` job porque a `trident-protect-controller-manager` ServiceAccount estava ausente ("Problema #1154").
- Correção das restaurações somente de PVC para incluírem uma `ownerReference` somente quando o proprietário existir.

O que há de novo na versão 26.06.2

Saiba o que há de novo no NetApp Trident e Trident Protect, incluindo melhorias, correções e descontinuações.

Trident

Melhorias

Correções

- **Kubernetes**:
  - Correção de um problema em que as regras da política de exportação do ONTAP-NAS eram excluídas silenciosamente quando o controlador do Trident era reiniciado ou uma `TridentBackendConfig` era atualizada enquanto `enableConcurrency` estava definido como `true`. Isso poderia deixar volumes já montados ("Problema #1166", "Problema #1179").

- Corrigimos as atualizações de backend que executavam uma verificação de órfãos por volume para alterações não relevantes, como credenciais e QoS, o que poderia bloquear as operações de CSI por horas em backends de grande porte. A verificação agora é executada somente quando o SVM ou o prefixo de storage é alterado.
- Correção da exibição de credenciais nos logs de depuração do Trident, incluindo despejos de solicitações gRPC do CSI, o despejo do ambiente de inicialização e `TridentBackendConfig` as especificações registradas pelo controlador CRD. O despejo do ambiente de inicialização agora registra valores apenas para as próprias `TRIDENT_*` variáveis do Trident e lista as outras variáveis por nome.

O que há de novo na versão 26.06.1

Saiba o que há de novo no NetApp Trident e Trident Protect, incluindo melhorias, correções e descontinuações.

Trident

Correções

- **Kubernetes**:
  - Corrigida uma condição de corrida entre a publicação CSI e a movimentação de volume que podia anexar um volume usando informações de acesso desatualizadas.
  - Corrigida uma condição de corrida em que NodeGetInfo poderia retornar rótulos de topologia vazios antes da conclusão do registro do nó, causando falhas no provisionamento com reconhecimento de topologia.
  - Corrigida uma rara condição de corrida de desconexão NVMe que podia derrubar um subsistema compartilhado enquanto outro pod ainda estava sendo conectado.
  - Corrigidas as falhas na publicação de clones de PVC NVMe (e o mapeamento incorreto do namespace pai) ao criar clones de um volume montado sob concorrência.
  - Corrigido o problema de falhas na criação de snapshots para PVCs em backends ONTAP-SAN-Economy usando um nameTemplate.

Melhorias

- **Kubernetes**:
  - Adicionada a opção `storageAPITimeout` (segundos) na configuração do backend ONTAP para definir os tempos limite dos clientes de API REST e ZAPI por backend.

O que há de novo em 26.06

Saiba o que há de novo no NetApp Trident e Trident Protect, incluindo melhorias, correções e descontinuações.

Trident

Melhorias

- **Kubernetes**:
  - Adicionado suporte de disponibilidade geral (GA) para simultaneidade nos drivers ONTAP-NAS-Economy e ONTAP-SAN-Economy, elevando essa funcionalidade da Tech Preview, e adicionado

suporte GA para simultaneidade nos drivers Azure NetApp Files (ANF) e SolidFire.

- Adicionado suporte para Kubernetes 1.36.
- Adicionada disponibilidade geral (GA) para volume com crescimento automático com os drivers NAS e SAN do Google Cloud NetApp Volumes (GCNV).
- Adicionada colocação balanceada do Unified ONTAP para seleção ideal de agregados ao provisionar novos volumes ("Problema #64").
- Adicionado suporte ONTAP AFX para o driver ONTAP-NAS-Flexgroup.
- Adicionada integração com o NetApp Shift Toolkit para simplificar as migrações de máquinas virtuais ao usar o Red Hat OpenShift Migration Toolkit for Virtualization.
- Adicionada a compatibilidade com Cloud Identity para o driver SAN do GCNV.
- Adicionado suporte a volumes subordinados para propagação do Trident Volume Publication (TVP) e fluxos de trabalho de volume com crescimento automático.
- Adicionado suporte para atualizações de volume e snapshots de grupo de volume ao executar Trident com a simultaneidade ativada.
- Adicionada limitação dinâmica de taxa gRPC para GCNV.
- Adicionada uma sonda de prontidão do controlador e ajustadas as configurações padrão de QPS e Burst da API do Kubernetes para o controlador e sidecars.
- Sequenciamento de operações de nós aprimorado ao separar as etapas de conexão, configuração de criptografia e formatação do sistema de arquivos.
- Os sistemas de arquivos raiz dos contêineres Trident foram montados como somente leitura para melhorar a segurança.
- Adicionada a remoção best-effort de snapshots para snapshots de volumes clonados em backends ONTAP quando o snapshot de origem de um volume clonado é excluído.
- Adicionada a funcionalidade de reconhecimento de PVCs de clones de backup para evitar que PVCs temporários criados por Veeam Kasten, Trilio e Cohesity fiquem presos na fila de recuperação de volumes do ONTAP.
- Melhoria no tratamento de ARNs da AWS para oferecer suporte a regiões especiais.
- Impedir a redução do tamanho dos Flexvols durante a operação de redimensionamento para o driver ONTAP-SAN-ECONOMY.

Melhorias experimentais

Não deve ser usado em ambientes de produção.

- **[Prévia Técnica]**: Adicionado suporte para importação de qtree ONTAP-NAS-Economy.
- **[Prévia Técnica]**: adicionado suporte para simultaneidade para os drivers ASA-R2 (SAN e NVMe).
- **[Prévia Técnica]**: Adicionado transporte HTTP GCNV ONTAP-mode para os drivers ONTAP-SAN e ONTAP-NAS.

Correções

- **Kubernetes**:
  - Corrigidas as condições de corrida e o gerenciamento de estado para operações simultâneas de publicação/despublicação, clonagem, cache e atualização de backend.

- Corrigidas as condições de corrida na política de exportação e o tratamento simultâneo de publicação/despublicação para volumes subordinados e clones somente leitura nos drivers ONTAP-NAS e ONTAP-NAS-Economy.
- Correção da descoberta do ONTAP-NAS-Economy FlexVol após um failover do MetroCluster quando as políticas de snapshot diferem ("Problema #1082").
- Corrigidas as novas tentativas de exclusão de volume ONTAP-NAS-Economy que interrompiam exclusões de qtree de longa duração ("Problema #1121").
- Corrigido o comportamento de importação e redimensionamento do ONTAP-SAN e ONTAP-SAN-Economy, incluindo validação de fsType, tratamento de metadados de volume e comportamento do modo de crescimento automático.
- Corrigida uma condição de corrida de namespace NVMe/TCP durante a criação simultânea de volumes ("Problema #1089").
- Corrigimos falhas no redimensionamento iSCSI que ocorriam quando os caminhos e portais disponíveis não correspondiam e melhoramos o comportamento em redimensionamentos repetidos.
- Corrigidas as falhas na exclusão de volumes ONTAP-SAN causadas por condições de volume ocupado.
- Correção no tratamento de erros REST do ONTAP e melhoria no comportamento de tempo limite do estado SVM.
- Correção no tratamento de pools zonais e grupos de hosts do GCNV SAN, normalização da conversão de identificadores de volume e normalização de solicitações de criação abaixo do tamanho mínimo.
- Correção na seleção do pool de capacidade nativo do GCNV para excluir pools no modo ONTAP.
- Correção do failover do KubeVirt para VMs que utilizam Trident Automated Workload Failover para acelerar o failover de VMs.
- Corrigidos problemas de RBAC em instalações gerenciadas pelo operador, incluindo ausência de ClusterRoles de nó e reconciliação incorreta de ClusterRole e ClusterRoleBinding durante as atualizações. Isso se aplica a instalações baseadas em Operator e Helm que usam o chart Trident Operator.
- Corrigidas diferenças persistentes de GitOps no TridentConfigurator e CRDs relacionados, omitindo valores de prioridade padrão e corrigindo a indentação do template YAML ("Problema #915", "Problema #1120").
- Correção na aplicação de patches em TridentVolumePublication CRDs durante uma atualização do Helm ("Problema #1151").
- Corrigida a ordem de inicialização do CSI, iniciando o serviço gRPC do CSI antes das novas tentativas de registro do nó.
- Corrigidas as sobreposições de configuração de clonagem SMB segura durante operações simultâneas.
- Corrigidas as anotações ausentes no pod do controlador Trident após uma atualização ("Problema #1004").
- Aprimorado o provisionamento lento de qtree em SVMs grandes ("Problema #1142").
- Corrigido ControllerPublish para usar o tipo de sistema de arquivos de configuração de volume em vez de um tipo de sistema de arquivos padrão incorreto.
- Corrigido um problema em que dispositivos dm órfãos podiam causar falhas repetidas de CSI NodeUnstageVolume.
- Melhorada a limpeza de nós obsoletos e multipath, removendo artefatos fantasmas de multipath e recursos TridentNode obsoletos após a limpeza de nós excluídos de forma lógica.

Trident Protect

Melhorias

- Adicionada a integração do OpenShift Service CA Operator para automatizar o gerenciamento de certificados TLS para o webhook de admissão do Trident Protect. Habilite definindo `useOpenshiftServiceCA` como `true` em seus valores do Helm. Para saber mais, consulte "Personalize a instalação do Trident Protect".
- Adicionado suporte para backup e restauração de VMs KubeVirt na OpenShift Virtualization. Proteja todas as VMs em um namespace ou direcione VMs específicas pelo nome e restaure seletivamente VMs individuais a partir de um backup ou snapshot. Para saber mais, consulte "Proteja máquinas virtuais KubeVirt com Trident Protect".
- Adicionado suporte para transformações de recursos, permitindo que você adicione, remova e modifique atributos de recursos do Kubernetes durante o processo de restauração. Para saber mais, consulte "Modifique os recursos à medida que os restaura usando recursos personalizados".
- Adicionado suporte para `storageClassMapping` em AppMirrorRelationship (AMR), permitindo o mapeamento de classe de armazenamento para PVCs individuais dentro de uma única definição de AMR. Para saber mais, consulte "Configure e gerencie a replicação de aplicativos usando o Trident Protect".

Correções

- Corrigido o problema de falha na restauração de máquina virtual entre clusters, que ocorria com um erro de webhook de admissão quando o namespace do aplicativo de origem não existia no cluster de destino.
- Corrigido o problema de falha na restauração seletiva de máquina virtual no local durante a etapa de criação do aplicativo devido a uma incompatibilidade de especificações com um recurso personalizado de aplicativo existente; o aplicativo existente agora é reutilizado.
- Corrigido o travamento de tarefas de restauração no local de máquina virtual em estado `Running` quando tanto a máquina virtual quanto seu PersistentVolumeClaim eram excluídos; a ausência de um PersistentVolumeClaim agora é tratada como um erro fatal.
- Corrigido o problema em que a edição de uma programação de proteção existente acionava incorretamente um backup ou snapshot imediato.
- Corrigido o problema em que o finalizador de snapshot tentava buscar o recurso personalizado do aplicativo após ele ter sido excluído; agora, o finalizador ignora a tentativa quando o aplicativo não é encontrado.
- Capacidade de storage protegida corrigida aumentando significativamente após uma restauração no local de máquina virtual, utilizando cópia esparsa durante a etapa de cópia do PersistentVolumeClaim.
- Corrigido o problema em que o Trident Protect excluía o objeto de snapshot do Kubernetes antes de confirmar a exclusão do snapshot subjacente no ONTAP; agora, o finalizador aguarda a `VolumeSnapshot` exclusão.
- Corrigido o problema em que os snapshots não prosseguiam quando uma operação de congelamento de máquina virtual falhava; as operações de congelamento e descongelamento agora continuam mesmo em caso de falha.
- Corrigido o problema de aplicativos de máquinas virtuais KubeVirt que recriavam incorretamente PersistentVolumeClaims durante as operações de reversão, ressincronização reversa e ressincronização de AppMirrorRelationship.

O que há de novo na versão 26.02.1

Saiba o que há de novo no NetApp Trident e Trident Protect, incluindo melhorias, correções e

descontinuações.

Trident

Correções

- **Kubernetes**:
- Corrigido o comportamento do SAN do Google Cloud NetApp Volumes (GCNV) para oferecer suporte a pools ZONAL do Flex Unified para cargas de trabalho em bloco (iSCSI). Agora, as configurações zonais e regionais são suportadas.
- Corrigido o tratamento do pool NAS UNIFIED do Google Cloud NetApp Volumes (GCNV) através da melhoria das operações de criação, validação, nomenclatura e pesquisa de volumes alternativos quando `InternalID` está ausente.
- Corrigido um problema em que as operações de redimensionamento de dispositivos iSCSI podiam falhar quando os caminhos e portais disponíveis não correspondiam.
- Corrigido um problema intermitente que `blkid` identificava incorretamente o tipo de sistema de arquivos para dispositivos LUKS, impedindo a criação de novos volumes.
- Corrigido um problema em que operações de publicação e preparação LUKS em grande escala podiam bloquear o controlador Trident. Agora, os usuários devem manter o controle das senhas LUKS associadas aos volumes.
- Corrigido um problema em que partições multipath iSCSI podiam aparecer como dispositivos fantasmas, impedindo que operações CSI `NodeUnstage` fossem bem-sucedidas.
- Corrigida uma condição de corrida no namespace NVMe/TCP durante a criação simultânea de volumes (Issue #1089).
- Corrigidas as operações simultâneas de clonagem de snapshots quando snapshots eram executados no mesmo segundo.
- Corrigido um problema que impedia as VMs KubeVirt usando `dataVolumes` de realizarem failover com Trident Automated Workload Failover.
- Corrigido um erro que ocorria ao excluir um recurso personalizado inválido `TridentMirrorRelationship` ao usar o núcleo concorrente do Trident.
- Correção da indentação na `tridentactionmirrorupdates`definição YAML do CRD do Kubernetes (Issue #1120).
- Corrigida a chamada da API REST de exclusão assíncrona do qtree (Problema nº 1121).
- Corrigido `ControllerPublish` para usar o tipo de sistema de arquivos de configuração de volume em vez de um tipo de sistema de arquivos padrão incorreto.
- Corrigido o registro de inicialização do controlador para relatar corretamente o status do driver simultâneo.
- Correção no tratamento de backends suspensos no Trident concorrente.
- Melhoria no tratamento de ARNs da AWS para oferecer suporte a regiões especiais.

Trident Protect

Correções

Dependências do Trident Protect atualizadas para corrigir vulnerabilidades de segurança:

- gRPC-Go: "CVE-2026-33186"

- Go x/crypto SSH: "CVE-2024-45337"
- OpenTelemetry-Go: "CVE-2026-24051"
- Tempo de execução Go: "CVE-2025-22871"
- Dependências adicionais: "CVE-2025-23109"

O que há de novo em 26.02

Saiba o que há de novo no NetApp Trident e Trident Protect, incluindo melhorias, correções e descontinuações.

Trident

Melhorias

- **Kubernetes**:
  - Adicionada disponibilidade geral (GA) para suporte à simultaneidade para ONTAP-NAS (somente NFS), ONTAP-SAN (iSCSI, FCP, NVMe) e Google Cloud NetApp Volumes (GCNV) drivers, graduando essa funcionalidade da versão de pré-visualização técnica. Consulte a "Documentação sobre escalabilidade do controlador" para mais informações.
  - Adicionado suporte para volume com crescimento automático no Trident com base em políticas AutoGrow do Trident personalizadas e definidas pelo usuário. Consulte a "Documentação de expansão automática de volume" para mais informações.
  - Concorrência de nós Trident aprimorada para melhorar a escalabilidade das operações de nós para volumes NVMe. Veja a "Documentação de volumes NVMe" para mais informações.
  - Adicionado suporte para o armazenamento automático em camadas de volumes NAS do GCNV por meio de anotações de PVC (`tieringPolicy`, `tieringMinimumCoolingDays`), incluindo seleção de pool e herança de clones. Veja a "Configure o armazenamento em camadas automático para a documentação do Google Cloud NetApp Volumes" para mais informações.
  - Adicionado suporte para volumes de bloco SAN GCNV (iSCSI) usando o `google-cloud-netapp-volumes-san` driver, incluindo provisionamento, mapeamento de grupo de hosts por nó para acesso a LUN e operações de clonagem a partir de volume. Consulte a"Documentação de configuração de blocos do Google Cloud NetApp Volumes" para mais informações.
  - Adicionado suporte para configuração automática de backend para Amazon FSx for NetApp ONTAP. Quando você cria um StorageClass que inclui os parâmetros necessários, Trident cria automaticamente o backend correspondente e o VolumeSnapshotClass, se necessário. Veja o "Documentação de configuração do backend do Amazon FSx for NetApp ONTAP" para mais informações.
  - Adicionado suporte para diferentes nuvens do Microsoft Azure, como Azure Government e Azure China, e configuração personalizada de nuvem para backends do Azure NetApp Files. Veja a "Documentação de configuração do backend do Azure NetApp Files" para mais informações.
  - Adicionado suporte para Kubernetes 1.35. Veja a "Documentação de requisitos" para mais informações.

Melhorias experimentais

> Não deve ser usado em ambientes de produção.

- **[Prévia Técnica]**: adicionado suporte para simultaneidade para os drivers ONTAP-NAS-Economy e

ONTAP-SAN-Economy.

Problemas conhecidos

> Existe um problema conhecido com o `csi-snapshotter` sidecar. Em todas as versões do Kubernetes, VolumeGroupSnapshots `v1beta1` impede que VolumeSnapshots alcancem o `ReadyToUse` estado.

Duas soluções alternativas estão disponíveis:

1. Exclua os CRDs VolumeGroupSnapshots para desativar VolumeGroupSnapshots e, em seguida, reinstale Trident.
2. Instale VolumeGroupSnapshots `v1beta2` e `snapshot-controller` versão 8.4.0 ou posterior e, em seguida, reinstale Trident. VolumeGroupSnapshots não funciona em versões do Kubernetes anteriores à v1.34.

Correções

- **Kubernetes**:
  - Corrigido um problema em que a remoção da publicação de um clone somente leitura removia as regras de política de exportação do volume de origem nos drivers ONTAP-NAS, ONTAP-NAS-Economy e Google Cloud NetApp Volumes "Problema #1086".
  - As imagens do kubectl foram alteradas para variantes leves baseadas em Alpine para evitar falhas de pull após a descontinuação pública de imagens pela Bitnami "Problema #1080".
  - Corrigida a preservação das anotações da implantação existente durante a atualização do Trident "Problema #1004".
  - Permitir clonagem entre diferentes classes de armazenamento se ambas as classes de armazenamento apontarem para o mesmo backend "Problema #1104".
  - Corrigidas falhas na preparação de nós causadas por timeouts em ambientes de nuvem com latências de rede. Aumentados os valores de timeout para instalações baseadas em nuvem.
  - Corrigido um problema na criação de LUN que fazia com que o atributo de tipo de sistema de arquivos permanecesse não definido quando o processo entrava em estado de nova tentativa.
  - Corrigida a pesquisa de volume da API REST para ignorar o estado do volume, evitando falsos negativos durante as consultas de volume.
  - Eficiência aprimorada do controlador Trident para o driver ontap-nas-economy quando usado em larga escala.
  - Definir internalID durante a importação de LUN no driver ontap-san-economy.
  - Aumentamos os limites de consulta do Azure Resource Graph para lidar com mais subnets.
  - Melhorados os tempos limite de divisão de clones CSI e ONTAP para evitar condições de corrida com alguns aplicativos de backup "Problema #1098", "Problema #1100".
  - Correção na supressão de mensagens de erro do LUKS "Problema #1069".
  - Correção no tratamento de mapeadores LUKS obsoletos para os protocolos iSCSI e NVMe. A lógica de limpeza aprimorada evita falhas de montagem causadas por mapeadores de dispositivos órfãos.
- Limitações de escala corrigidas para volumes RWX NVMe.
- Atualizado o pacote OpenTelemetry-Go para corrigir "CVE-2026-24051".

Trident Protect

Melhorias

- Trident Protect agora desativa automaticamente os agendamentos de proteção e cancela as operações em andamento antes de uma restauração no local, e os reativa após a conclusão da restauração. Para saber mais, consulte "Restaure aplicativos usando o Trident Protect.".
- Adicionado o campo `runImmediately` ao schedule CR e o sinalizador CLI `--run-immediately` para acionar um backup ou snapshot imediato após a criação do agendamento. Para saber mais, consulte "Criar um cronograma de proteção de dados".
- Adicionada a possibilidade de especificar um nome personalizado para a aplicação restaurada usando o `destinationApplicationName` campo no CR de restauração ou o `--destination-app-name` parâmetro da CLI. Para saber mais, consulte "Restaure aplicativos usando o Trident Protect.".

Correções

- Corrigidas falhas na restauração causadas pela criação de pods antes que as contas de serviço necessárias estivessem disponíveis.
- Corrigido `Roles`e `RoleBindings` sendo ignorados durante as restaurações de aplicativos.
- Corrigido o problema em que o nome do cluster de origem não era exibido na saída `tridentctl-protect get appvaultcontent`, apesar de estar configurado corretamente.
- Corrigidos erros de restauração do Kopia que eram ignorados devido à falta de `pipefail` tratamento.
- Corrigidas falhas em snapshots e backup causadas por filtros de recursos que excluíam volumes persistentes.
- Corrigida a restauração incorreta de PVC em aplicações com múltiplos namespaces que possuíam PVC com nomes idênticos em diferentes namespaces, o que poderia resultar em perda de dados.

O que há de novo em 25.10

Saiba mais sobre as novidades do Trident e do Trident Protect, incluindo melhorias, correções e recursos descontinuados.

Trident

Melhorias

- **Kubernetes**:
  - Adicionado suporte para snapshots de grupos de volumes CSI com as APIs Kubernetes v1beta1 para snapshots de grupos de volumes para os drivers ONTAP-NAS NFS e ONTAP-SAN-Economy, além do ONTAP-SAN (iSCSI e FC). Ver"Trabalhar com instantâneos de grupos de volumes" .
  - Adicionado suporte para failover automático de cargas de trabalho com desanexação forçada de volumes para os drivers ONTAP-NAS e ONTAP-NAS-Economy (excluindo SMB em ambos os drivers NAS), e para os drivers ONTAP-SAN e ONTAP-SAN-Economy. Ver"Automatizando o failover de aplicações com estado usando o Trident." .
  - Concorrência aprimorada de nós Trident para maior escalabilidade nas operações de nós para volumes FCP.
  - Adicionado suporte a ONTAP AFX para o driver ONTAP NAS. Ver"Exemplos e opções de configuração do ONTAP nas" .

- Adicionado suporte para configurar solicitações e limites de recursos de CPU e memória para contêineres Trident por meio de valores de CR (Create Request) e Helm chart do TridentOrchestrator. ("Edição nº 1000" ,"Edição nº 927" ,"Edição nº 853" ,"Edição nº 592" ,"Edição nº 110").
- Adicionado suporte a FC para a personalidade ASAr2. Ver"Exemplos e opções de configuração de SAN ONTAP" .
- Adicionada uma opção para disponibilizar métricas do Prometheus com HTTPS, em vez de HTTP. Ver"Monitore o Trident" .
- Adicionada uma opção `--no-rename` Ao importar um volume, o objetivo é manter o nome original, mas permitir que o Trident gerencie o ciclo de vida do volume. Ver"Importar volumes" .
- A implantação do Trident agora é executada na classe de prioridade system-cluster-critical.

- Adicionada uma opção para o controlador Trident usar a rede do host via helm, operator e tridentctl ("Edição nº 858").
- Adicionado suporte manual de QoS ao driver ANF, tornando-o pronto para produção no Trident 25.10; esse aprimoramento experimental foi introduzido no Trident 25.06.

Melhorias experimentais

Não deve ser usado em ambientes de produção.

- **[Prévia técnica]**: Adicionado suporte para simultaneidade para ONTAP-NAS (somente NFS) e ONTAP-SAN (NVMe para ONTAP 9 unificado), além da prévia técnica existente para o driver ONTAP-SAN (protocolos iSCSI e FCP no ONTAP 9 unificado).

Correções

- **Kubernetes**:
  - Corrigida a inconsistência no nome do contêiner CSI node-driver-registrar, padronizando o DaemonSet do Linux para node-driver-registrar, a fim de corresponder à nomenclatura do DaemonSet do Windows e da imagem do contêiner.
  - Corrigido um problema em que as políticas de exportação para qtrees legados não eram atualizadas corretamente.
- **Openshift**:
  - Corrigido o problema que impedia a inicialização do pod do nó Trident em nós Windows no Openshift devido ao SCC ter o allowHostDirVolumePlugin definido como falso ("Edição nº 950").
- Corrigido o problema em que o QPS da API do Kubernetes não estava sendo configurado via Helm ("Edição nº 975").
- Corrigida a impossibilidade de montar uma Persistent Volume Claim (PVC) baseada em um snapshot de uma PVC de sistema de arquivos XFS baseado em NVMe no mesmo nó do Kubernetes.
- Corrigido o problema de alteração do UUID após a reinicialização do host/Docker no modo NDVP, adicionando nomes de subsistema exclusivos/compartilhados por backend (por exemplo, netappdvp_subsystem).
- Corrigidos erros de montagem para volumes iSCSI durante a atualização do Trident de versões anteriores à 23.10 para a 24.10 e superiores, resolvendo o problema de "SANType inválido".
- Corrigido o problema em que o estado do backend do Trident não estava sendo atualizado para online/offline sem reiniciar o controlador do Trident .
- Corrigida condição de corrida intermitente que causava redimensionamento lento do tubo de PVC.

- Corrigido o problema em que os snapshots não eram apagados em caso de falha na clonagem de volumes.
- Corrigido o problema que impedia a remoção do volume da fase de preparação quando o caminho do dispositivo era alterado pelo kernel.
- Corrigido o problema que impedia a descompactação do volume devido ao dispositivo LUKS já estar fechado.
- Corrigido problema em que operações de armazenamento lentas causavam erros de ContextDeadline.
- O Trident Operator aguardará o tempo limite configurável do Kubernetes (k8s-timeout) para verificar a versão do Trident .

Trident Protect

O NetApp Trident Protect oferece recursos avançados de gerenciamento de dados de aplicativos que aprimoram a funcionalidade e a disponibilidade de aplicativos Kubernetes com estado, com suporte dos sistemas de armazenamento NetApp ONTAP e do provisionador de armazenamento NetApp Trident CSI.

Melhorias

- Adicionadas anotações para controlar os tempos limite de CR de snapshot para CRs de agendamento e backup:
  - `protect.trident.netapp.io/snapshot-completion-timeout`
  - `protect.trident.netapp.io/volume-snapshots-ready-to-use-timeout`
  - `protect.trident.netapp.io/volume-snapshots-created-timeout`

  Ver"Suporte para backup e anotações de agendamento." .
- Adicionada uma anotação ao CR de agendamento para configurar o tempo limite de vinculação do PVC, que será usado pelo CR de backup: `protect.trident.netapp.io/pvc-bind-timeout-sec` . Ver"Suporte para backup e anotações de agendamento." .
- Melhorou `tridentctl-protect` Listagens de backup e snapshot com um novo campo para indicar falhas no gancho de execução.

Mudanças em 25.06.2

Trident

Correções

- **Kubernetes**: Corrigido um problema crítico em que dispositivos iSCSI incorretos eram descobertos ao desanexar volumes de nós do Kubernetes.

Mudanças em 25.06.1

Trident

> Para clientes que utilizam SolidFire, não atualizem para a versão 25.06.1 devido a um problema conhecido ao cancelar a publicação de volumes. A versão 25.06.2 será lançada em breve para corrigir esse problema.

Correções

- **Kubernetes**:
  - Corrigido um problema em que os NQNs não eram verificados antes de serem desmapeados dos subsistemas.
  - Corrigido um problema em que várias tentativas de fechar um dispositivo LUKS levavam a falhas na desanexação de volumes.
  - Corrigido o problema de desempacotamento do volume iSCSI quando o caminho do dispositivo foi alterado desde sua criação.
  - Clonagem de blocos de volumes em classes de armazenamento.
- **OpenShift**: Corrigido um problema em que a preparação do nó iSCSI falhava com o OCP 4.19.
- Aumentou o tempo limite ao clonar um volume usando backends SolidFire ("Problema nº 1008").

Mudanças em 25.06

Trident

Melhorias

- **Kubernetes**:
  - Adicionado suporte para instantâneos de grupo de volume CSI com `v1beta1` Snapshot do grupo de volumes das APIs do Kubernetes para o driver ONTAP-SAN iSCSI. "Trabalhar com instantâneos de grupos de volumes"Consulte .

    > VolumeGroupSnapshot é um recurso beta do Kubernetes com APIs beta. O Kubernetes 1.32 é a versão mínima necessária para o VolumeGroupSnapshot.

  - Adicionado suporte para ONTAP ASA r2 para NVMe/TCP além de iSCSI. Ver"Exemplos e opções de configuração de SAN ONTAP" .
  - Adicionado suporte SMB seguro para volumes ONTAP-NAS e ONTAP-NAS-Economy. Usuários e grupos do Active Directory agora podem ser usados com volumes SMB para maior segurança. "Habilitar SMB seguro"Consulte .
  - Simultaneidade de nós Trident aprimorada para maior escalabilidade em operações de nós para volumes iSCSI.
  - Adicionado `--allow-discards` ao abrir volumes LUKS para permitir comandos discard/TRIM para recuperação de espaço.
  - Desempenho aprimorado ao formatar volumes criptografados com LUKS.
  - Limpeza LUKS aprimorada para dispositivos LUKS com falha, mas parcialmente formatados.
  - Idempotência de nó Trident aprimorada para conexão e desanexação de volume NVMe.
  - Adicionado `internalID` campo para a configuração do volume Trident para o driver ONTAP-SAN-Economy.
  - Adicionado suporte para replicação de volume com SnapMirror para backends NVMe. "Replique volumes usando o SnapMirror"Consulte .

Melhorias experimentais

> Não deve ser usado em ambientes de produção.

- [Visualização técnica] Habilitou operações simultâneas do controlador Trident por meio do `--enable-concurrency` sinalizador de recurso. Isso permite que as operações do controlador sejam executadas em paralelo, melhorando o desempenho em ambientes grandes ou movimentados.

  > Este recurso é experimental e atualmente suporta fluxos de trabalho paralelos limitados com o driver ONTAP-SAN (protocolos iSCSI e FCP).

- [Visualização técnica] Adicionado suporte manual de QOS com o driver ANF.

Correções

- **Kubernetes**:
  - Corrigido um problema com CSI NodeExpandVolume em que dispositivos multipath podiam ficar com tamanhos incongruentes quando discos SCSI subjacentes não estavam disponíveis.
  - Falha corrigida na limpeza de políticas de exportação duplicadas para drivers ONTAP-NAS e ONTAP-NAS-Economy.
  - Corrigido o problema em que os volumes GCNV usavam NFSv3 por padrão quando `nfsMountOptions` não estava definido; agora, ambos os protocolos NFSv3 e NFSv4 são suportados. Se `nfsMountOptions` não for fornecido, a versão de NFS padrão do host (NFSv3 ou NFSv4) será usada.
  - Problema de implantação corrigido ao instalar o Trident usando o Kustomize ("Problema nº 831").
  - Políticas de exportação ausentes corrigidas para PVCs criados a partir de instantâneos ("Problema nº 1016").
  - Problema corrigido em que os tamanhos de volume ANF não eram alinhados automaticamente em incrementos de 1 GiB.
  - Problema corrigido ao usar NFSv3 com Bottlerocket.
- Problema corrigido com volumes ONTAP-NAS-Economy expandindo até 300 TB apesar de falhas de redimensionamento.
- Problema corrigido em que as operações de divisão de clones estavam sendo feitas de forma síncrona ao usar a API REST do ONTAP.

Depreciações:

- **Kubernetes**: Kubernetes mínimo suportado atualizado para v1.27.

Trident Protect

O NetApp Trident Protect oferece recursos avançados de gerenciamento de dados de aplicativos que aprimoram a funcionalidade e a disponibilidade de aplicativos Kubernetes com estado, com suporte dos sistemas de armazenamento NetApp ONTAP e do provisionador de armazenamento NetApp Trident CSI.

Melhorias

- Tempos de restauração melhorados, oferecendo a opção de fazer backups completos mais frequentes.
- Granularidade aprimorada da definição do aplicativo e restauração seletiva com filtragem Grupo-Versão-

Tipo (GVK).

- Ressincronização eficiente e replicação reversa ao usar o AppMirrorRelationship (AMR) com o NetApp SnapMirror, para evitar a replicação completa de PVC.
- Adicionada a capacidade de usar o EKS Pod Identity para criar buckets do AppVault, eliminando a necessidade de especificar um segredo com as credenciais do bucket para clusters EKS.
- Adicionada a capacidade de pular a restauração de rótulos e anotações no namespace de restauração, se necessário.
- O AppMirrorRelationship (AMR) agora verificará a expansão do PVC de origem e executará a expansão apropriada no PVC de destino, conforme necessário.

Correções

- Corrigido o bug em que os valores de anotação de snapshots anteriores estavam sendo aplicados a snapshots mais recentes. Todas as anotações de snapshots agora são aplicadas corretamente.
- Definiu um segredo para criptografia do movimentador de dados (Kopia / Restic) por padrão, se não definido.
- Adicionadas mensagens de validação e erro aprimoradas para a criação do S3 AppVault.
- O AppMirrorRelationship (AMR) agora replica apenas PVs no estado Bound, para evitar tentativas malsucedidas.
- Problema corrigido em que erros eram exibidos ao obter AppVaultContent em um AppVault com grande número de backups.
- Os KubeVirt VMSnapshots são excluídos das operações de restauração e failover para evitar falhas.
- Problema corrigido com o Kopia em que os snapshots estavam sendo removidos prematuramente devido ao cronograma de retenção padrão do Kopia substituir o que foi definido pelo usuário no cronograma.

Mudanças em 25.02.1

Trident

Correções

- **Kubernetes**:
  - Corrigido um problema no operador Trident em que nomes e versões de imagens sidecar eram incorretamente preenchidos ao usar um Registro de imagem não padrão ("Problema nº 983").
  - Corrigido o problema em que as sessões multipath não recuperavam durante um failover do ONTAP ("Problema nº 961").

Mudanças em 25,02

A partir da versão 25.02 do Trident , o resumo "Novidades" fornece detalhes sobre melhorias, correções e itens descontinuados para as versões do Trident e do Trident Protect.

Trident

Melhorias

- **Kubernetes**:
  - Adicionado suporte para ONTAP ASA R2 para iSCSI.

- Adicionado suporte para Force Detach para volumes ONTAP-nas durante cenários de encerramento de nó não gracioso. Os novos volumes ONTAP-nas agora utilizarão políticas de exportação por volume gerenciadas pelo Trident. Forneceu um caminho de atualização para que os volumes existentes façam a transição para o novo modelo de política de exportação na não publicação, sem afetar os workloads ativos.
- Adicionada anotação cloneFromSnapshot.
- Adicionado suporte para clonagem de volume entre namespace.
- Correcções de verificação melhoradas de recuperação automática iSCSI para iniciar redigitalizações por ID de host, canal, destino e LUN exato.
- Adicionado suporte para Kubernetes 1,32.

- **OpenShift**:
  - Adicionado suporte para preparação automática de nó iSCSI para RHCOS em clusters ROSA.
  - Adicionado suporte para virtualização OpenShift para drivers ONTAP.
- Adicionado suporte Fibre Channel no driver ONTAP-SAN.
- Adicionado suporte a NVMe LUKS.
- Mudou para imagem de raspadinha para todas as imagens base.
- Foi adicionada a deteção e o registo do estado da ligação iSCSI quando as sessões iSCSI devem ser efetuadas, mas não são ("Problema nº 961").
- Adicionado suporte a volumes SMB com o driver google-Cloud-NetApp-volumes.
- Adicionado suporte para permitir que os volumes ONTAP saltem a fila de recuperação na eliminação.
- Adicionado suporte para substituir imagens padrão usando Shas em vez de tags.
- Adicionado sinalizador image-pull-segies para o instalador tridentctl.

Correções

- **Kubernetes**:
  - Corrigido endereços IP de nó ausentes das políticas de exportação automática ("Problema nº 965").
  - Políticas de exportação automáticas fixas alternando prematuramente para política de volume por ONTAP-nas-Economy.
  - Credenciais de configuração de back-end fixas para oferecer suporte a todas as partições ARN () da AWS disponíveis"Problema nº 913".
  - Opção adicionada para desativar a reconciliação do configurador automático no operador Trident ("Problema nº 924").
  - Adicionado securityContext para o contentor csi-Resizer ("Problema nº 976").

Trident Protect

O NetApp Trident Protect oferece recursos avançados de gerenciamento de dados de aplicativos que aprimoram a funcionalidade e a disponibilidade de aplicativos Kubernetes com estado, com suporte dos sistemas de armazenamento NetApp ONTAP e do provisionador de armazenamento NetApp Trident CSI.

Melhorias

- Adicionado suporte para backup e restauração de VMs de virtualização KubeVirt / OpenShift para armazenamento com volumeMode: File e volumeMode: Block (dispositivo bruto). Este suporte é

compatível com todos os drivers Trident e aprimora os recursos de proteção existentes ao replicar o armazenamento usando o NetApp SnapMirror com Trident Protect.

- Adicionada a capacidade de controlar o comportamento de congelamento no nível da aplicação para ambientes Kubevirt.
- Adicionado suporte para configurar conexões proxy AutoSupport.
- Adicionada a capacidade de definir um segredo para a criptografia do controlador de dados (Kopia / Restic).
- Adicionado a capacidade de executar manualmente um gancho de execução.
- Adicionada a capacidade de configurar restrições de contexto de segurança (SCCs) durante a instalação do Trident Protect.
- Adicionada a possibilidade de configurar o nodeSelector durante a instalação do Trident Protect.
- Adicionado suporte para proxy de saída HTTP / HTTPS para objetos AppVault.
- Extended ResourceFilter para habilitar a exclusão de recursos com escopo de cluster.
- Adicionado suporte para o token de sessão da AWS nas credenciais do S3 AppVault.
- Adicionado suporte para coleta de recursos após ganchos de execução pré-snapshot.

Correções

- Melhorou o gerenciamento de volumes temporários para ignorar a fila de recuperação de volume do ONTAP.
- As anotações SCC são agora restauradas para os valores originais.
- Eficiência de restauração aprimorada com suporte para operações paralelas.
- Suporte aprimorado para hook timeouts de execução para aplicativos maiores.

Mudanças em 24.10.1

Melhorias

- **Kubernetes**: Adicionado suporte ao Kubernetes 1,32.
- Foi adicionada a deteção e o registo do estado da ligação iSCSI quando as sessões iSCSI devem ser efetuadas, mas não são ("Problema nº 961").

Correções

- Corrigido endereços IP de nó ausentes das políticas de exportação automática ("Problema nº 965").
- Políticas de exportação automáticas fixas alternando prematuramente para política de volume por ONTAP-nas-Economy.
- Dependências do Trident e do Trident-ASUP atualizadas para endereçar CVE-2024-45337 e CVE-2024-45310.
- Logouts removidos para portais não CHAP não-CHAP intermitentemente insalubres durante a auto-recuperação iSCSI ("Problema nº 961").

Mudanças em 24,10

Melhorias

- O driver do Google Cloud NetApp volumes agora está disponível para volumes NFS e é compatível com provisionamento com reconhecimento de zona.
- O código de carga de trabalho do GCP será usado como o Cloud Identity para volumes do Google Cloud NetApp com o GKE.
- Adicionado `formatOptions` parâmetro de configuração aos drivers ONTAP-SAN e ONTAP-SAN-Economy para permitir que os usuários especifiquem opções de formato LUN.
- Tamanho mínimo de volume Azure NetApp Files reduzido para 50 GiB. O novo tamanho mínimo do Azure deverá estar disponível em novembro.
- Parâmetro de configuração adicionado `denyNewVolumePools` para restringir drivers de economia ONTAP-nas e economia ONTAP-SAN a pools FlexVol pré-existentes.
- Adição, remoção ou renomeação de agregados do SVM em todos os drivers ONTAP.
- Adicionou 18 MiB de sobrecarga aos LUNs LUKS para garantir que o tamanho de PVC relatado seja utilizável.
- Estágio de nó ONTAP-SAN e ONTAP-SAN aprimorado e manipulação de erros de desinstalação para permitir a remoção do estágio para remover dispositivos após um estágio com falha.
- Adicionado um gerador de funções personalizado, permitindo que os clientes criem um papel minimalista para o Trident no ONTAP.
- Adicionado registo adicional para resolução de problemas `lsscsi` ("Problema nº 792").

Kubernetes

- Adição de novos recursos do Trident para workflows nativos do Kubernetes:
  - Proteção de dados
  - Migração de dados
  - Recuperação de desastres
  - Mobilidade de aplicativos

  "Saiba mais sobre o Trident Protect.".
- Adicionou uma nova bandeira `--k8s-api-qps` aos instaladores para definir o valor QPS usado pelo Trident para se comunicar com o servidor da API do Kubernetes.
- Sinalizador adicionado `--node-prep` aos instaladores para gerenciamento automático de dependências de protocolo de storage nos nós de cluster do Kubernetes. Compatibilidade testada e verificada com o protocolo de armazenamento iSCSI do Amazon Linux 2023
- Adicionado suporte para forçar desanexar para volumes de economia de ONTAP-nas durante cenários de encerramento de nó não gracioso.
- Os novos volumes de NFS com economia de ONTAP nas usarão políticas de exportação por qtree ao usar `autoExportPolicy` a opção de back-end. As Qtrees só serão mapeadas para políticas de exportação restritivas de nós no momento da publicação para melhorar o controle de acesso e a segurança. Os qtrees existentes serão alternados para o novo modelo de política de exportação quando o Trident não publicar o volume de todos os nós para fazê-lo sem afetar cargas de trabalho ativas.
- Adicionado suporte para Kubernetes 1,31.

Melhorias experimentais

- Adicionado pré-visualização técnica para suporte de Fibre Channel no driver ONTAP-SAN.

Correções

- **Kubernetes**:
  - Webhook de admissão de Rancher fixo que impede instalações de Helm do Trident ("Problema nº 839").
  - Tecla de afinidade fixa nos valores do gráfico de leme ("Problema nº 898").
  - Corrigido tridentControllerPluginNodeSeletor/tridentNodePluginNodeSeletor não funcionará com o valor "verdadeiro" ("Problema nº 899").
  - Instantâneos efêmeros eliminados criados durante a clonagem ("Problema nº 901").
- Adicionado suporte para o Windows Server 2019.
- Corrigido 'go mod tidy' em Trident repo ("Problema nº 767").

Desvalorizações

- **Kubernetes:**
  - Mínimo atualizado com suporte de Kubernetes para 1,25.
  - Suporte removido para a Diretiva de Segurança DO POD.

Rebranding do produto

A partir do lançamento de 24,10, o Astra Trident é renomeado para Trident (NetApp Trident). Esse rebranding não afeta recursos, plataformas suportadas ou interoperabilidade para o Trident.

Mudanças em 24,06

Melhorias

- **IMPORTANTE**: O `limitVolumeSize` parâmetro agora limita os tamanhos de qtree/LUN nos drivers ONTAP Economy. Use o novo `limitVolumePoolSize` parâmetro para controlar tamanhos de FlexVol nesses drivers. ("Problema nº 341").
- Adicionada capacidade de recuperação automática iSCSI para iniciar varreduras SCSI por ID LUN exato se grupos obsoletos estiverem em uso ("Problema nº 883").
- Adicionado suporte para operações de clone de volume e redimensionamento para ser permitido mesmo quando o back-end está no modo suspenso.
- Adicionada capacidade para que as configurações de log configuradas pelo usuário para o controlador Trident sejam propagadas para pods de nó do Trident.
- Adicionado suporte no Trident para usar REST por padrão em vez de ONTAPI (ZAPI) para ONTAP versões 9.15.1 e posteriores.
- Adicionado suporte a metadados e nomes de volume personalizados nos back-ends de storage do ONTAP para novos volumes persistentes.
- Aprimorado o `azure-netapp-files` driver (ANF) para habilitar automaticamente o diretório snapshot por padrão quando as opções de montagem NFS estão definidas para usar a versão 4.x.
- Adicionado suporte de Bottlerocket para volumes NFS.

- Adicionado suporte a pré-visualização técnica para o Google Cloud NetApp volumes.

Kubernetes

- Adicionado suporte para Kubernetes 1,30.
- Adicionado capacidade para Trident DaemonSet para limpar montagens de Zumbis e arquivos de rastreamento residuais na inicialização ("Problema nº 883").
- Adicionada anotação em PVC `trident.netapp.io/luksEncryption` para importar dinamicamente volumes LUKS ("Problema nº 849").
- Adição de reconhecimento de topologia para o driver do ANF.
- Adicionado suporte para nós do Windows Server 2022.

Correções

- Falhas de instalação do Trident fixas devido a transações obsoletas.
- Corrigido o tridentctl para ignorar mensagens de aviso do Kubernetes ("Problema nº 892").
- A prioridade do controlador Trident foi alterada `SecurityContextConstraint` para `0` ("Problema nº 887").
- Os drivers ONTAP agora aceitam tamanhos de volume abaixo de 20 MiB ("Problema[n.o 885").
- Corrigido Trident para evitar a redução de volumes FlexVol durante a operação de redimensionamento para o driver ONTAP-SAN.
- Falha fixa de importação de volume do ANF com NFS v4,1.

Mudanças em 24,02

Melhorias

- Adicionado suporte para o Cloud Identity.
  - AKS com ANF - o Azure Workload Identity será usado como identidade de nuvem.
  - O EKS com FSxN - função do AWS IAM será usado como identidade na nuvem.
- Adicionado suporte para instalar o Trident como um complemento no cluster EKS a partir do console EKS.
- Adicionada capacidade de configurar e desativar a recuperação automática iSCSI ("Problema nº 864").
- A personalidade do Amazon FSX foi adicionada aos drivers do ONTAP para permitir a integração com o AWS IAM e o SecretsManager e permitir que o Trident exclua volumes do FSX com backups ("Problema nº 453").

Kubernetes

- Adicionado suporte para Kubernetes 1,29.

Correções

- Mensagens de aviso do ACP fixas, quando o ACP não está ativado ("Problema nº 866").
- Adicionado um atraso de 10 segundos antes de executar uma divisão de clones durante a exclusão de snapshot para drivers ONTAP, quando um clone está associado ao snapshot.

Desvalorizações

- Estrutura de atestações in-toto removida dos manifestos de imagem multi-plataforma.

Mudanças em 23,10

Correções

- Expansão de volume fixa se um novo tamanho solicitado for menor do que o tamanho total do volume para os drivers de armazenamento ONTAP-nas e ONTAP-nas-FlexGroup ("Problema nº 834").
- Tamanho de volume fixo para exibir somente o tamanho utilizável do volume durante a importação para drivers de armazenamento ONTAP-nas e ONTAP-nas-FlexGroup ("Problema nº 722").
- Conversão de nomes FlexVol fixos para ONTAP-nas-Economy.
- Corrigido problema de inicialização do Trident em um nó do Windows quando o nó é reinicializado.

Melhorias

Kubernetes

Adicionado suporte para Kubernetes 1,28.

Trident

- Adicionado suporte para o uso de identidades gerenciadas do Azure (AMI) com o driver de armazenamento azure-NetApp-Files.
- Adicionado suporte para NVMe sobre TCP para o driver ONTAP-SAN.
- Adicionada capacidade de pausar o provisionamento de um volume quando o back-end é definido como estado suspenso pelo usuário ("Problema nº 558").

Mudanças em 23.07.1

Kubernetes: exclusão do daemonset fixa para oferecer suporte a atualizações sem inatividade ("Problema nº 740").

Mudanças em 23,07

Correções

Kubernetes

- Atualização do Trident corrigida para ignorar pods antigos presos no estado de terminação ("Problema nº 740").
- Adicionado tolerância à definição "transient-Trident-version-pod" ("Problema nº 795").

Trident

- Solicitações ONTAPI (ZAPI) fixas para garantir que os números de série LUN sejam consultados ao obter atributos LUN para identificar e corrigir dispositivos iSCSI fantasma durante as operações de estadiamento do nó.
- Corrigido o erro de manipulação no código do driver de armazenamento ("Problema nº 816").

- Ajuste o tamanho da cota ao usar drivers ONTAP com o uso-REST.
- Criação de clone de LUN fixo em ONTAP-san-Economy.
- Reverter campo de informações de publicação `rawDevicePath` de para `devicePath`; lógica adicionada para preencher e recuperar (em alguns casos) `devicePath` campo.

Melhorias

Kubernetes

- Adicionado suporte para importar instantâneos pré-provisionados.
- Implementação minimizada e permissões do daemonset linux ("Problema nº 817").

Trident

- Não é mais relatar o campo de estado para volumes e instantâneos "online".
- Atualiza o estado de back-end se o back-end do ONTAP estiver off-line ("Problemas nº 801", "Nº 543").
- O número de série LUN é sempre recuperado e publicado durante o fluxo de trabalho ControllerVolumePublish.
- Adicionada lógica adicional para verificar o número de série e o tamanho do dispositivo multipath iSCSI.
- Verificação adicional para volumes iSCSI para garantir que o dispositivo multipath correto seja desorganizado.

Aperfeiçoamento experimental

Adicionado suporte de visualização técnica para NVMe sobre TCP para o driver ONTAP-SAN.

Documentação

Muitas melhorias organizacionais e de formatação foram feitas.

Desvalorizações

Kubernetes

- Suporte removido para instantâneos v1beta1.
- Suporte removido para volumes pré-CSI e classes de armazenamento.
- Mínimo atualizado com suporte de Kubernetes para 1,22.

Mudanças em 23,04

> Forçar a desagregação de volume para volumes ONTAP-SAN-* é compatível apenas com versões Kubernetes com o recurso desativação de nó não-gracioso ativado. Forçar a desligação deve ser ativada no momento da instalação utilizando o `--enable-force-detach` sinalizador do instalador do Trident.

Correções

- Operador Trident fixo para usar localhost IPv6 para instalação quando especificado na especificação.
- Permissões de função de cluster do operador do Trident fixas para serem sincronizadas com as

permissões do pacote ("Problema nº 799").

- Corrigido o problema com a inclusão de volume de bloco bruto em vários nós no modo RWX.
- Suporte fixo à clonagem de FlexGroup e importação de volume para volumes SMB.
- Corrigido o problema em que o controlador Trident não podia desligar imediatamente ("Problema nº 811").
- Correção adicionada para listar todos os nomes do grupo igrop associados a um LUN especificado provisionado com drivers ONTAP-San-*.
- Adicionada uma correção para permitir que processos externos sejam executados até a conclusão.
- Corrigido erro de compilação para a arquitetura s390 ("Problema nº 537").
- Corrigido o nível de registo incorreto durante as operações de montagem de volume ("Problema nº 781").
- Corrigido erro de afirmação de tipo potencial ("Problema nº 802").

Melhorias

- Kubernetes:
  - Adicionado suporte para Kubernetes 1,27.
  - Adicionado suporte para importar volumes LUKS.
  - Adicionado suporte para o modo de acesso ao PVC ReadWriteOncePod.
  - Adicionado suporte para Force Detach para volumes ONTAP-SAN-* durante cenários de encerramento de nó não gracioso.
  - Todos os volumes ONTAP-SAN-* agora usarão grupos por nó. Os LUNs só serão mapeados para os grupos enquanto forem publicados ativamente nesses nós para melhorar a nossa postura de segurança. Os volumes existentes serão oportunisticamente comutados para o novo esquema de grupos quando o Trident determinar que é seguro fazê-lo sem afetar cargas de trabalho ativas ("Problema nº 758").
  - Melhor segurança do Trident ao limpar grupos não utilizados gerenciados pelo Trident dos backends ONTAP-SAN-*.
- Adicionado suporte para volumes SMB com o Amazon FSX para os drivers de armazenamento ONTAP-nas-Economy e ONTAP-nas-FlexGroup.
- Adicionado suporte para compartilhamentos SMB com os drivers de storage ONTAP-nas, ONTAP-nas-Economy e ONTAP-nas-FlexGroup.
- Adicionado suporte para arm64 nós ("Problema nº 732").
- Procedimento de encerramento aprimorado do Trident desativando primeiro os servidores API ("Problema nº 811").
- Adicionado suporte de compilação entre plataformas para Windows e hosts arm64 para Makefile; veja BUILD.md.

Desvalorizações

Kubernetes: Os grupos com escopo de back-end não serão mais criados ao configurar drivers ONTAP-san e ONTAP-san-Economy ("Problema nº 758").

Mudanças em 23.01.1

Correções

- Operador Trident fixo para usar localhost IPv6 para instalação quando especificado na especificação.
- Permissões fixas da função de cluster do operador do Trident para estar em sincronia com as permissões do pacote "Problema nº 799".
- Adicionada uma correção para permitir que processos externos sejam executados até a conclusão.
- Corrigido o problema com a inclusão de volume de bloco bruto em vários nós no modo RWX.
- Suporte fixo à clonagem de FlexGroup e importação de volume para volumes SMB.

Mudanças em 23,01

O Kubernetes 1,27 agora é compatível com o Trident. Atualize o Trident antes de atualizar o Kubernetes.

Correções

- Kubernetes: Adicionadas opções para excluir a criação da Diretiva de Segurança do Pod para corrigir instalações do Trident via Helm ("Problemas nº 783, nº 794").

Melhorias

Kubernetes

- Adicionado suporte para Kubernetes 1,26.
- Utilização geral aprimorada de recursos RBAC do Trident ("Problema nº 757").
- Automação adicionada para detetar e corrigir sessões iSCSI quebradas ou obsoletas em nós de host.
- Adicionado suporte para expandir volumes criptografados LUKS.
- Kubernetes: Suporte à rotação de credenciais adicionado para volumes criptografados LUKS.

Trident

- Adicionado suporte para volumes SMB com o Amazon FSX for NetApp ONTAP para o driver de armazenamento ONTAP-nas.
- Adicionado suporte para permissões NTFS ao usar volumes SMB.
- Adicionado suporte a pools de storage para volumes do GCP com nível de serviço CVS.
- Adicionado suporte para uso opcional do flexgroupAggregateList ao criar FlexGroups com o driver de armazenamento ONTAP-nas-FlexGroup.
- Desempenho aprimorado para o driver de storage econômico ONTAP nas ao gerenciar vários volumes FlexVol
- Atualizações de dataLIF habilitadas para todos os drivers de storage nas do ONTAP.
- Atualização da convenção de nomenclatura Trident Deployment e DaemonSet para refletir o sistema operacional do nó host.

Desvalorizações

- Kubernetes: Mínimo atualizado com suporte de Kubernetes para 1,21.
- DataLIFs não devem mais ser especificados ao configurar `ontap-san` ou `ontap-san-economy` drivers.

Mudanças em 22,10

Você deve ler as seguintes informações críticas antes de atualizar para o Trident 22,10.

** informações críticas sobre o Trident 22.10 **

- O Kubernetes 1,25 agora é compatível com o Trident. É necessário atualizar o Trident para o 22,10 antes da atualização para o Kubernetes 1,25.
- O Trident agora reforça estritamente o uso de configuração multipathing em ambientes SAN, com um valor recomendado de `find_multipaths: no` no arquivo multipath.conf.

  O uso de configuração não multipathing ou o uso `find_multipaths: yes` de ou `find_multipaths: smart` valor no arquivo multipath.conf resultará em falhas de montagem. A Trident recomenda o uso de `find_multipaths: no` desde a versão 21,07.

Correções

- Corrigido um problema específico para o back-end do ONTAP criado usando `credentials` campo que não aparece on-line durante a atualização do 22.07.0 ("Problema nº 759").
- **Docker:** corrigiu um problema que fazia com que o plugin de volume do Docker não iniciasse em alguns ambientes ("Problema nº 548" e "Problema nº 760").
- Corrigido problema de SLM específico para backends de SAN ONTAP para garantir que apenas subconjunto de dataLIFs pertencentes a nós de relatório seja publicado.
- Corrigido problema de desempenho em que verificações desnecessárias para iSCSI LUNs aconteceram ao anexar um volume.
- Novas tentativas granulares removidas dentro do fluxo de trabalho iSCSI do Trident para falhar rapidamente e reduzir os intervalos de tentativas externas.
- Corrigido o problema em que um erro foi retornado ao lavar um dispositivo iSCSI quando o dispositivo multipath correspondente já estava lavado.

Melhorias

- Kubernetes:
  - Adicionado suporte para Kubernetes 1,25. É necessário atualizar o Trident para o 22,10 antes da atualização para o Kubernetes 1,25.
  - Adicionado um ServiceAccount separado, ClusterRole e ClusterRoleBinding para a implantação do Trident e DaemonSet para permitir melhorias futuras de permissões.
  - Adicionado suporte para "compartilhamento de volume entre namespace".
- Todos os drivers de storage Trident `ontap-*` agora funcionam com a API REST do ONTAP.
- Adicionado novo operador yaml (`bundle_post_1_25.yaml`) sem um `PodSecurityPolicy` para oferecer suporte ao Kubernetes 1,25.
- Adicionado "Suporte para volumes criptografados com LUKS" para `ontap-san` e `ontap-san-economy` drivers de armazenamento.
- Adicionado suporte para nós do Windows Server 2019.
- Adicionado "Suporte para volumes SMB em nós do Windows" através do `azure-netapp-files` driver de armazenamento.
- A deteção automática de comutação MetroCluster para controladores ONTAP está agora disponível em

geral.

Desvalorizações

- **Kubernetes:** atualizado com o mínimo de Kubernetes compatível para 1,20.
- Driver do Astra Data Store (ADS) removido.
- Removido o suporte `yes` e `smart` as opções para `find_multipaths` quando configurar multipathing de nó de trabalho para iSCSI.

Mudanças em 22,07

Correções

Kubernetes

- Corrigido problema para lidar com valores booleanos e numéricos para o seletor de nó ao configurar o Trident com Helm ou o Operador Trident. ("GitHub Edição nº 700")
- Corrigido problema no tratamento de erros do caminho não-CHAP, de modo que kubelet irá tentar novamente se falhar. "GitHub Edição nº 736")

Melhorias

- Transição do k8s.gcr.io para o registry.k8s.io como Registro padrão para imagens CSI
- Os volumes ONTAP-SAN agora usarão grupos por nó e mapearão apenas LUNs para grupos enquanto são publicados ativamente nesses nós para melhorar nossa postura de segurança. Os volumes existentes serão oportunisticamente comutados para o novo esquema de grupos quando o Trident determinar que é seguro fazê-lo sem afetar cargas de trabalho ativas.
- Incluído um ResourceQuota com instalações Trident para garantir que o Trident DaemonSet seja programado quando o consumo de PriorityClass é limitado por padrão.
- Adicionado suporte para recursos de rede ao driver Azure NetApp Files. ("GitHub Edição nº 717")
- Adicionada deteção automática de comutação MetroCluster de pré-visualização técnica aos drivers ONTAP. ("GitHub Edição nº 228")

Desvalorizações

- **Kubernetes:** atualizado com o mínimo de Kubernetes compatível para 1,19.
- A configuração de backend não permite mais vários tipos de autenticação em uma única configuração.

Remoções

- O driver do AWS CVS (obsoleto desde 22,04) foi removido.
- Kubernetes
  - Removido recurso SYS_ADMIN desnecessário dos pods de nós.
  - Reduz o nodeprep para informações simples de host e descoberta de serviço ativo para confirmar o melhor esforço de que os serviços NFS/iSCSI estão disponíveis nos nós de trabalho.

Documentação

Uma nova "Padrões de segurança do pod"seção (PSS) foi adicionada detalhando as permissões habilitadas pelo Trident na instalação.

Mudanças em 22,04

A NetApp está continuamente melhorando e aprimorando seus produtos e serviços. Aqui estão alguns dos recursos mais recentes do Trident. Para versões anteriores, "Versões anteriores da documentação" consulte .

> Se você estiver atualizando de qualquer versão anterior do Trident e usar o Azure NetApp Files, o `location` parâmetro config agora é um campo único obrigatório.

Correções

- Análise melhorada de nomes de iniciadores iSCSI. ("GitHub Edição nº 681")
- Corrigido problema em que os parâmetros da classe de armazenamento CSI não eram permitidos. ("GitHub Edição nº 598")
- Declaração de chave duplicada corrigida no CRD Trident. ("GitHub Edição nº 671")
- Corrigidos registos de instantâneos do CSI imprecisos. ("GitHub Edição nº 629"))
- Corrigido o problema com a remoção de volumes em nós excluídos. ("GitHub Edição nº 691")
- Adição de manipulação de inconsistências de sistema de arquivos em dispositivos de bloco. ("GitHub Edição nº 656")
- Corrigido problema ao puxar imagens de suporte automático ao definir o `imageRegistry` sinalizador durante a instalação. ("GitHub Edição nº 715")
- Corrigido o problema em que o driver Azure NetApp Files não conseguiu clonar um volume com várias regras de exportação.

Melhorias

- As conexões de entrada para os endpoints seguros da Trident agora exigem um mínimo de TLS 1,3. ("GitHub Edição nº 698")
- O Trident agora adiciona cabeçalhos HSTS às respostas de seus endpoints seguros.
- O Trident agora tenta ativar o recurso de permissões unix do Azure NetApp Files automaticamente.
- **Kubernetes**: O daemonset do Trident agora é executado na classe de prioridade crítica do nó do sistema. ("GitHub Edição nº 694")

Remoções

O driver da série e (desativado desde 20,07) foi removido.

Mudanças em 22.01.1

Correções

- Corrigido o problema com a remoção de volumes em nós excluídos. ("GitHub Edição nº 691")
- Corrigido o pânico ao acessar campos nil para espaço agregado nas respostas da API do ONTAP.

Mudanças em 22.01.0

Correções

- **Kubernetes:** aumente o tempo de repetição do backoff do Registro de nós para clusters grandes.
- Corrigido problema em que o driver azure-NetApp-Files poderia ser confundido por vários recursos com o mesmo nome.
- Os DataLIFs SAN IPv6 da ONTAP agora funcionam se forem especificados com colchetes.
- Corrigido o problema em que a tentativa de importar um volume já importado retorna EOF deixando PVC em estado pendente. ("GitHub Edição nº 489")
- Corrigido o problema quando o desempenho do Trident diminui quando > 32 snapshots são criados em um volume SolidFire.
- Substituído SHA-1 por SHA-256 na criação de certificado SSL.
- Driver Azure NetApp Files fixo para permitir nomes de recursos duplicados e limitar as operações a um único local.
- Driver Azure NetApp Files fixo para permitir nomes de recursos duplicados e limitar as operações a um único local.

Melhorias

- Melhorias do Kubernetes:
  - Adicionado suporte para Kubernetes 1,23.
  - Adicione opções de agendamento para pods Trident quando instalado via Operador Trident ou Helm. ("GitHub Edição nº 651")
- Permitir volumes entre regiões no driver do GCP. ("GitHub Edição nº 633")
- Adicionado suporte para a opção 'unixPermissions' para volumes Azure NetApp Files. ("GitHub Edição nº 666")

Desvalorizações

A interface REST do Trident pode ouvir e servir apenas em endereços 127.0.0.1 ou [::1]

Mudanças em 21.10.1



A versão v21.10.0 tem um problema que pode colocar o controlador Trident em um estado CrashLoopBackOff quando um nó é removido e depois adicionado de volta ao cluster do Kubernetes. Esse problema foi corrigido no v21,10.1 (GitHub Issue 669).

Correções

- Condição de corrida potencial fixa ao importar um volume em um back-end CVS do GCP, resultando em falha na importação.
- Corrigido um problema que pode colocar o controlador Trident em um estado CrashLoopBackOff quando um nó é removido e depois adicionado de volta ao cluster do Kubernetes (problema 669 do GitHub).
- Corrigido o problema em que os SVMs não eram mais descobertos se nenhum nome SVM foi especificado (problema 612 do GitHub).

Mudanças em 21.10.0

Correções

- Corrigido o problema em que clones de volumes XFS não podiam ser montados no mesmo nó que o volume de origem (problema 514 do GitHub).
- Corrigido o problema em que o Trident registrou um erro fatal no desligamento (problema 597 do GitHub).
- Correções relacionadas ao Kubernetes:
  - Retorne o espaço usado de um volume como o mínimo restoresSize ao criar snapshots com `ontap-nas` drivers e `ontap-nas-flexgroup` (GitHub Issue 645).
  - Corrigido o problema em que `Failed to expand filesystem` o erro foi registrado após o redimensionamento de volume (GitHub problema 560).
  - Corrigido o problema em que um pod poderia ficar preso `Terminating` no estado (GitHub problema 572).
  - Corrigido o caso em que um `ontap-san-economy` FlexVol pode estar cheio de LUNs instantâneos (GitHub problema 533).
  - Corrigido o problema do instalador personalizado YAML com imagem diferente (problema 613 do GitHub).
  - Corrigido cálculo do tamanho do instantâneo (GitHub edição 611).
  - Corrigido o problema em que todos os instaladores do Trident podiam identificar o Kubernetes simples como OpenShift (problema 639 do GitHub).
  - Corrigido o operador do Trident para parar a reconciliação se o servidor da API do Kubernetes não estiver acessível (problema 599 do GitHub).

Melhorias

- Adicionado suporte à `unixPermissions` opção para volumes de performance do GCP-CVS.
- Adicionado suporte para volumes CVS otimizados para escala no GCP na faixa de 600 GiB a 1 TIB.
- Aprimoramentos relacionados ao Kubernetes:
  - Adicionado suporte para Kubernetes 1,22.
  - Habilitou o operador do Trident e o gráfico Helm para trabalhar com o Kubernetes 1,22 (GitHub Issue 628).
  - Adicionado a imagem do operador ao `tridentctl` comando imagens (GitHub Issue 570).

Melhorias experimentais

- Adicionado suporte para replicação de volume no `ontap-san` driver.
- Adicionado suporte REST **Tech Preview** para os `ontap-nas-flexgroup` drivers , `ontap-san`, e `ontap-nas-economy` .

Problemas conhecidos

Problemas conhecidos identificam problemas que podem impedi-lo de usar o produto com sucesso.

- Ao atualizar um cluster do Kubernetes do 1,24 para o 1,25 ou posterior que tenha o Trident instalado, você

deve atualizar o Values.yaml para definir `excludePodSecurityPolicy true` ou adicionar `--set excludePodSecurityPolicy=true helm upgrade` ao comando antes de atualizar o cluster.

- O Trident agora aplica um espaço em `fsType` (`fsType=""`branco) para volumes que não têm o `fsType` especificado em seu StorageClass. Ao trabalhar com o Kubernetes 1,17 ou posterior, a Trident dá suporte a fornecer um espaço em branco `fsType` para volumes NFS. Para volumes iSCSI, é necessário definir o `fsType` no StorageClass ao aplicar um `fsGroup` contexto de uso de segurança.
- Ao usar um back-end em várias instâncias do Trident, cada arquivo de configuração de back-end deve ter um valor diferente `storagePrefix` para backends do ONTAP ou usar um diferente `TenantName` para backends do SolidFire. O Trident não consegue detetar volumes criados por outras instâncias do Trident. Tentar criar um volume existente em backends ONTAP ou SolidFire é bem-sucedido, porque o Trident trata a criação de volume como uma operação idempotente. Se `storagePrefix` ou `TenantName` não forem diferentes, pode haver colisões de nomes para volumes criados no mesmo back-end.
- Ao instalar o Trident (usando `tridentctl` ou o Operador do Trident) e usar `tridentctl` para gerenciar o Trident, você deve garantir que a `KUBECONFIG` variável de ambiente esteja definida. Isso é necessário para indicar o cluster do Kubernetes com `tridentctl` quem trabalhar. Ao trabalhar com vários ambientes do Kubernetes, você deve garantir que o `KUBECONFIG` arquivo seja obtido com precisão.
- Para executar a recuperação de espaço on-line para PVS iSCSI, o SO subjacente no nó de trabalho pode exigir que as opções de montagem sejam passadas para o volume. Isso é verdade para instâncias RHEL/Red Hat Enterprise Linux CoreOS (RHCOS), que exigem o `discard` "opção de montagem"; Certifique-se de que a opção Descartar mountOption está incluída em sua[`StorageClass`lista] para suportar descarte de blocos on-line.
- Se você tiver mais de uma instância do Trident por cluster do Kubernetes, o Trident não poderá se comunicar com outras instâncias e não poderá descobrir outros volumes que eles criaram, o que leva a um comportamento inesperado e incorreto se mais de uma instância for executada em um cluster. Deve haver apenas uma instância do Trident por cluster do Kubernetes.
- Se objetos baseados em Trident `StorageClass` forem excluídos do Kubernetes enquanto o Trident estiver offline, o Trident não removerá as classes de armazenamento correspondentes de seu banco de dados quando ele voltar online. Você deve excluir essas classes de armazenamento usando `tridentctl` ou a API REST.
- Se um usuário excluir um PV provisionado pelo Trident antes de excluir o PVC correspondente, o Trident não excluirá automaticamente o volume de backup. Você deve remover o volume via `tridentctl` ou a API REST.
- A ONTAP não pode provisionar simultaneamente mais de um FlexGroup de cada vez, a menos que o conjunto de agregados seja exclusivo para cada solicitação de provisionamento.
- Ao usar o Trident sobre IPv6, você deve especificar `managementLIF` e `dataLIF` na definição de back-end entre colchetes. Por exemplo, `[fd20:8b1e:b258:2000:f816:3eff:feec:0]`.

> Não é possível especificar `dataLIF` em um back-end de SAN ONTAP. O Trident descobre todas as LIFs iSCSI disponíveis e as usa para estabelecer a sessão multipath.

- Se estiver usando `solidfire-san` o driver com OpenShift 4,5, certifique-se de que os nós de trabalho subjacentes usem MD5 como o algoritmo de autenticação CHAP. Os algoritmos CHAP seguros compatíveis com FIPS SHA1, SHA-256 e SHA3-256 estão disponíveis com o Element 12,7.

Encontre mais informações

- "Trident GitHub"

- "Trident blogs"

Versões anteriores da documentação

Se você não estiver executando Trident 26.10, a documentação para versões anteriores está disponível com base no "Ciclo de vida do suporte da Trident".

- "Trident 26.06"
- "Trident 26.02"
- "Trident 25.10"
- "Trident 25.06"
- "Trident 25,02"
- "Trident 24,10"
- "Trident 24,06"
- "Trident 24,02"
- "Trident 23,10"
- "Trident 23,07"
- "Trident 23,04"
- "Trident 23,01"

NetApp Trident suporte para sistemas de storage ONTAP ASA r2

NetApp Trident 25.02 e versões posteriores suportam sistemas NetApp ASA r2 como backends de armazenamento. Consulte "Sistemas ASA r2" para mais informações.

ASA r2 sistemas requerem o `ontap-san`driver. Trident não oferece suporte ao `ontap-san-economy`driver para sistemas ASA r2.

Ao especificar `ontap-san` como o `storageDriverName` na configuração do backend, o Trident detecta automaticamente o sistema de storage ASA r2.

Trident oferece proteção de dados limitada para sistemas ASA r2 com Trident protect.

Os protocolos SAN suportados dependem da sua versão do Trident:

- Trident 25.02 e versões posteriores suportam iSCSI.
- Trident 25.06 e versões posteriores suportam NVMe/TCP além de iSCSI.
- Trident 25.10 e versões posteriores suportam FC, além de iSCSI e NVMe/TCP.

Você deve atribuir pelo menos um agregado à máquina virtual de storage (SVM) para o storage de backend ONTAP. Consulte "Atribuindo agregados a SVMs em sistemas ASA r2" para obter instruções.

Operações suportadas

- Provisionamento de volumes persistentes (PVs)
- Provisionamento dinâmico de volumes
- Criando e excluindo volumes
- Clonando volumes
- Expansão de volumes
- Gerenciando classes de armazenamento

Operações não suportadas

- Criptografia do Linux Unified Key Setup (LUKS)
- SnapMirror volume replicação
- Limitação do uso agregado
- Modo de reserva de espaço
- Instantâneos
- Camadas

Para obter mais informações, consulte "Exemplos e opções de configuração de SAN ONTAP" .

Problemas conhecidos

Problemas conhecidos identificam problemas que podem impedi-lo de usar esta versão do produto com sucesso.

Os seguintes problemas conhecidos afetam a versão atual:

Os volumes permanecem anexados após a atualização do Trident 22.10 ou anterior

Este problema afeta clientes que atualizam diretamente do Trident 22.10 ou anterior para o Trident 26.10 com volumes anexados. Atualizações do Trident 23.01 ou posterior não são afetadas.

Após a atualização, os volumes afetados podem não se desanexar. Os pods que usam volumes ReadWriteOnce (RWO) não podem ser movidos para outro nó e podem relatar um erro de anexação múltipla. A exclusão do PersistentVolume (PV) também pode travar.

Para contornar esse problema, no nó que possui o anexo de volume obsoleto, exclua o `volumePublishInfo.json`arquivo remanescente do `globalmount`diretório desse volume.

VolumeSnapshots não atingindo o estado ReadyToUse



Existe um problema conhecido com o `csi-snapshotter` sidecar. Em todas as versões do Kubernetes, VolumeGroupSnapshots `v1beta1` impede que VolumeSnapshots alcancem o `ReadyToUse` estado.

Duas soluções alternativas estão disponíveis:

1. Exclua os CRDs VolumeGroupSnapshots para desativar VolumeGroupSnapshots e, em seguida, reinstale Trident.
2. Instale VolumeGroupSnapshots `v1beta2` e `snapshot-controller` versão 8.4.0 ou posterior e, em seguida, reinstale Trident. VolumeGroupSnapshots não funciona em versões do Kubernetes anteriores à v1.34.

Restaurar backups Restic de arquivos grandes pode falhar

Ao restaurar arquivos de 30 GB ou maiores a partir de um backup do Amazon S3 feito com o Restic, a operação de restauração pode falhar. Como solução alternativa, faça backup dos dados usando o Kopia como gerenciador de dados (o Kopia é o gerenciador de dados padrão para backups). Consulte "Proteja aplicativos usando o Trident Protect." para obter instruções.

Comece agora

Saiba mais sobre o Trident

Saiba mais sobre o Trident

O Trident é um projeto de código aberto totalmente suportado mantido pela NetApp. Ele foi desenvolvido para ajudar você a atender às demandas de persistência da sua aplicação em contêineres usando interfaces padrão do setor, como a Container Storage Interface (CSI).

O que é o Trident?

O NetApp Trident permite o consumo e o gerenciamento de recursos de storage em todas as plataformas de storage NetApp populares, na nuvem pública ou em ambientes locais, incluindo clusters ONTAP locais (AFF, FAS e ASA), ONTAP Select, Cloud Volumes ONTAP, Element software (NetApp HCI, SolidFire), Azure NetApp Files, Amazon FSx for NetApp ONTAP e Google Cloud NetApp Volumes.

O Trident é um orquestrador de storage dinâmico e em conformidade com a Container Storage Interface (CSI) que se integra nativamente ao "Kubernetes". O Trident é executado como um único Pod de controlador e um Pod de nó em cada nó de trabalho no cluster. "Arquitetura da Trident"Consulte para obter detalhes.

O Trident também fornece integração direta com o ecossistema do Docker para plataformas de storage NetApp. O plug-in de volume do Docker do NetApp (nDVP) dá suporte ao provisionamento e gerenciamento de recursos de storage da plataforma de storage para hosts do Docker. "Implante o Trident para Docker"Consulte para obter detalhes.

> Se esta for a primeira vez que usa o Kubernetes, você deve se familiarizar com o "Conceitos e ferramentas do Kubernetes".

Plataformas Kubernetes suportadas

Trident é compatível com uma variedade de distribuições e plataformas Kubernetes.

As plataformas suportadas incluem: * Kubernetes upstream * Red Hat OpenShift * SUSE Harvester 1.7.0 (ONTAP iSCSI)

Integração do Kubernetes com os produtos NetApp

O portfólio de produtos de storage do NetApp se integra a muitos aspectos de um cluster Kubernetes, fornecendo recursos avançados de gerenciamento de dados que melhoram o recurso, a funcionalidade, a performance e a disponibilidade da implantação do Kubernetes.

▼ **Amazon FSX para NetApp ONTAP**

"Amazon FSX para NetApp ONTAP" É um serviço AWS totalmente gerenciado que permite iniciar e executar sistemas de arquivos equipados com o sistema operacional de storage NetApp ONTAP.

▼ **Azure NetApp Files**

"Azure NetApp Files" É um serviço de compartilhamento de arquivos do Azure de nível empresarial, desenvolvido pela NetApp. É possível executar os workloads mais exigentes baseados em arquivos no Azure de forma nativa, com a performance e o gerenciamento de rich data que você espera do NetApp.

▼ **Cloud Volumes ONTAP**

"Cloud Volumes ONTAP" É um dispositivo de storage somente de software que executa o software de gerenciamento de dados ONTAP na nuvem.

▼ **Google Cloud NetApp volumes**

"Google Cloud NetApp volumes" O Google Cloud é um serviço de storage de arquivos totalmente gerenciado que oferece storage de arquivos de nível empresarial e de alta performance.

▼ **Software Element**

"Elemento" permite que o administrador de storage consolide workloads garantindo a performance e possibilitando um espaço físico do storage simplificado e otimizado.

▼ **NetApp HCI**

"NetApp HCI" simplifica o gerenciamento e a escala do data center automatizando tarefas de rotina e permitindo que os administradores de infraestrutura se concentrem em funções mais importantes.

O Trident pode provisionar e gerenciar dispositivos de storage para aplicações em contêiner diretamente na plataforma de storage subjacente da NetApp HCI.

▼ **NetApp ONTAP**

"NetApp ONTAP" É o sistema operacional de storage unificado multiprotocolo da NetApp que oferece recursos avançados de gerenciamento de dados para qualquer aplicação.

Os sistemas ONTAP têm configurações all-flash, híbridas ou totalmente HDD e oferecem muitos modelos de implantação diferentes: Clusters FAS, AFA e ASA no local, ONTAP Select e Cloud Volumes ONTAP. O Trident oferece suporte a esses modelos de implantação do ONTAP.

Arquitetura da Trident

O Trident é executado como um único Pod de controlador e um Pod de nó em cada nó de trabalho no cluster. O pod de nó deve estar em execução em qualquer host onde você queira montar um volume Trident potencialmente.

Compreensão dos pods dos nós e dos pods do controlador

O Trident é implantado como Pod do controlador Tridentum único e um ou mais Pods do nó Trident no cluster do Kubernetes e usa contentores Sidecar padrão do Kubernetes para simplificar a implantação de plug-ins do CSI. "Kubernetes CSI Sidecar contêineres" São mantidos pela comunidade do Kubernetes Storage.

Kubernetes "seletores de nós" e "tolerações e taints"são usados para restringir um pod a ser executado em um nó específico ou preferencial. Você pode configurar seletores de nós e tolerações para pods de nó e controlador durante a instalação do Trident.

- O plugin controlador lida com o provisionamento e gerenciamento de volume, como snapshots e redimensionamento.
- O plug-in do nó manipula a conexão do armazenamento ao nó.

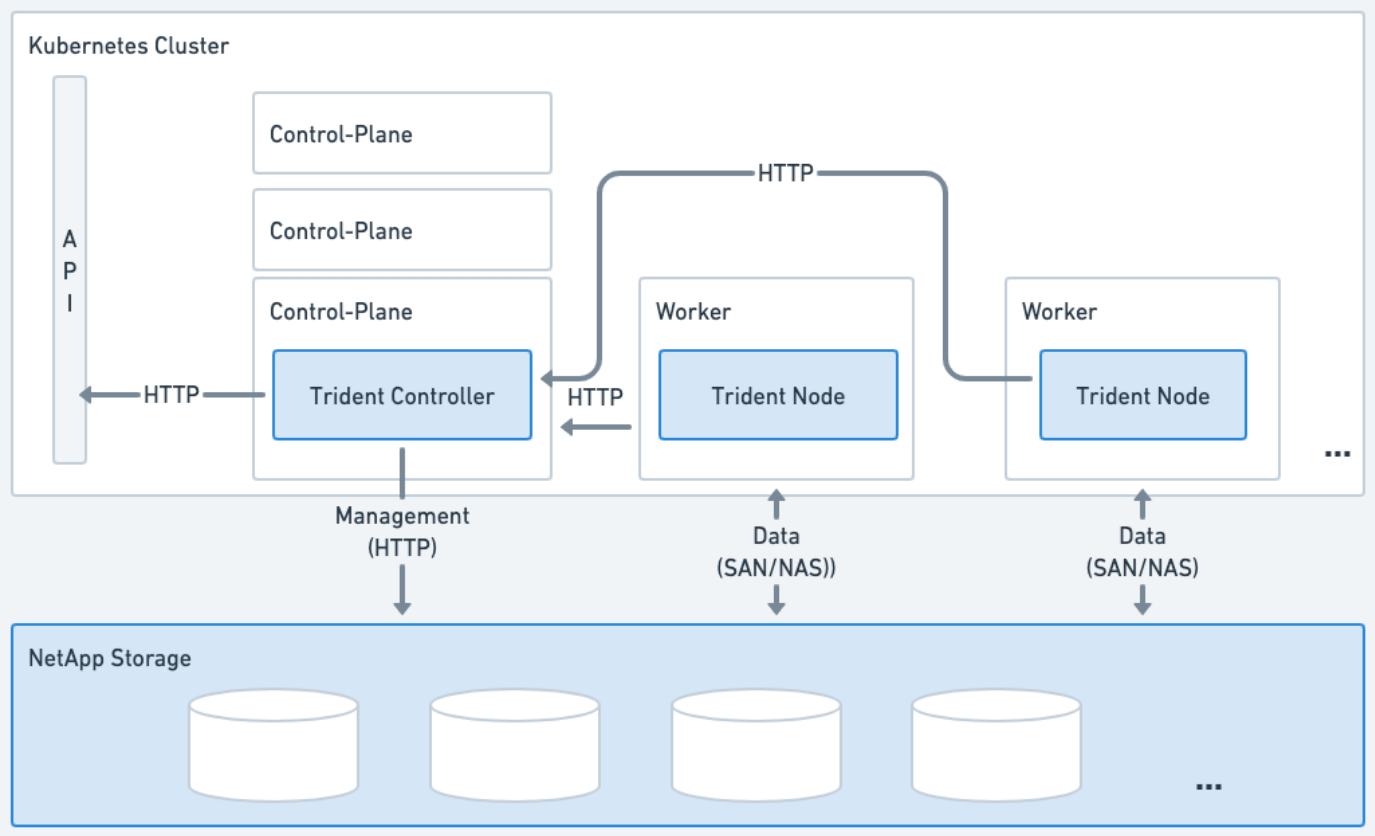


Figura 1. A Trident implantou no cluster do Kubernetes

Pod do controlador Trident

O Pod do controlador Trident é um único Pod que executa o plug-in do controlador CSI.

- Responsável pelo provisionamento e gerenciamento de volumes no storage NetApp
- Gerenciado por uma implantação do Kubernetes
- Pode ser executado no plano de controle ou nos nós de trabalho, dependendo dos parâmetros de instalação.

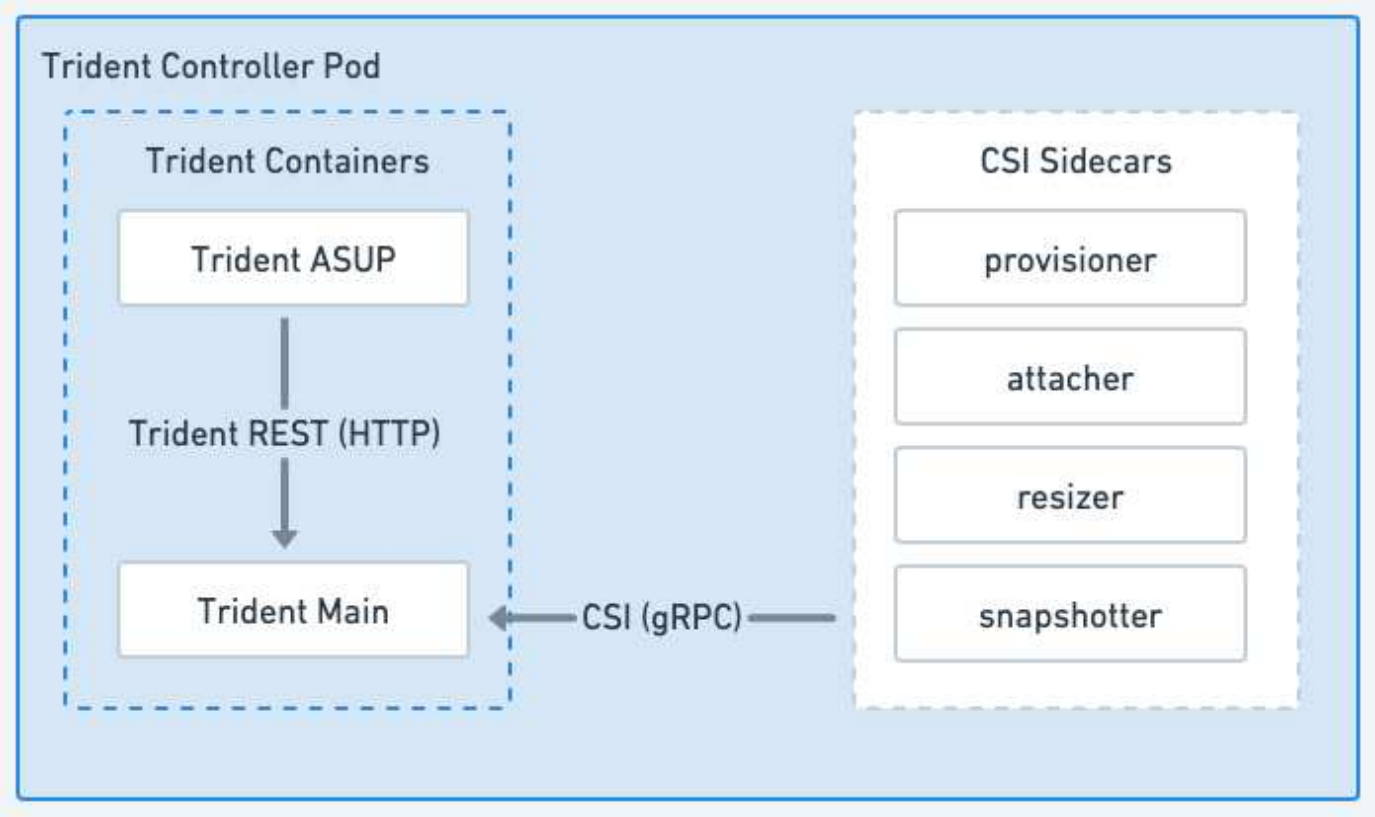


Figura 2. Diagrama do pod do controlador Trident

Pods do nó Trident

Os pods de nó Trident são pods privilegiados que executam o plug-in do nó CSI.

- Responsável pela montagem e desmontagem do armazenamento dos pods em execução no host
- Gerenciado por um DaemonSet Kubernetes
- Deve ser executado em qualquer nó que montar o storage NetApp

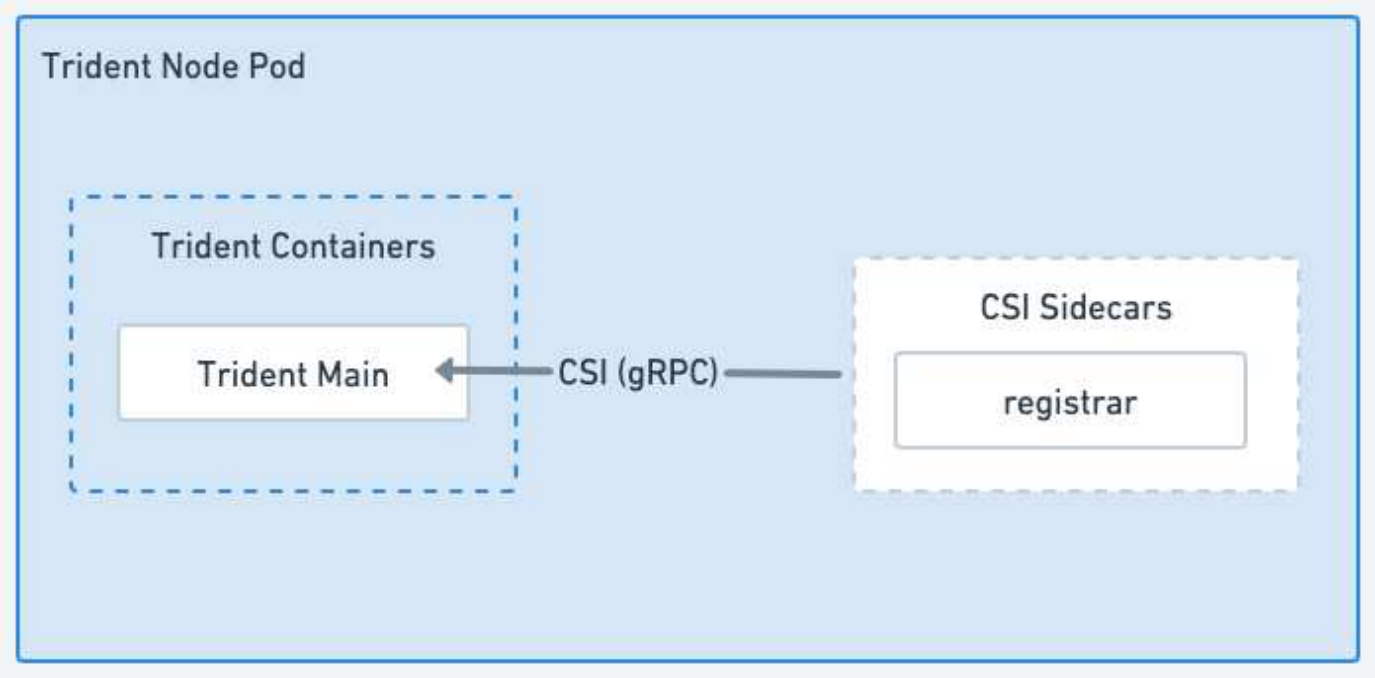


Figura 3. Diagrama do pod do nó Trident

Arquiteturas de cluster Kubernetes compatíveis

O Trident é compatível com as seguintes arquiteturas do Kubernetes:

Arquiteturas de cluster do Kubernetes	Suportado	Instalação predefinida
Único mestre, computação	Sim	Sim
Mestre múltiplo, computação	Sim	Sim
Mestre `etcd`, , computação	Sim	Sim
Mestre, infraestrutura, computação	Sim	Sim

Conceitos

Processo de provisionamento de volume do Trident

O provisionamento no Trident tem duas fases principais. A primeira fase associa uma classe de armazenamento ao conjunto de conjuntos de armazenamento de back-end adequados e ocorre como uma preparação necessária antes do provisionamento. A segunda fase inclui a própria criação de volume e requer a escolha de um pool de armazenamento daqueles associados à classe de armazenamento do volume pendente.

Associação de classe de armazenamento

A associação de pools de storage de back-end a uma classe de armazenamento depende dos atributos solicitados da classe de armazenamento e `storagePools` das listas , `additionalStoragePools` e `excludeStoragePools` . Quando você cria uma classe de storage, o Trident compara os atributos e pools oferecidos por cada um de seus back-ends aos solicitados pela classe de storage. Se os atributos e o nome de um pool de armazenamento corresponderem a todos os atributos e nomes de pool solicitados, o Trident adicionará esse pool de armazenamento ao conjunto de pools de armazenamento adequados para essa

classe de armazenamento. Além disso, o Trident adiciona todos os pools de storage listados na `additionalStoragePools` lista a esse conjunto, mesmo que seus atributos não preencham todos ou nenhum dos atributos solicitados da classe de armazenamento. Você deve usar a `excludeStoragePools` lista para substituir e remover pools de armazenamento de uso para uma classe de armazenamento. O Trident executa um processo semelhante toda vez que você adiciona um novo back-end, verificando se seus pools de armazenamento atendem aos das classes de armazenamento existentes e removendo quaisquer que tenham sido marcados como excluídos.

Criação de volume

Em seguida, o Trident usa as associações entre classes de armazenamento e pools de armazenamento para determinar onde provisionar volumes. Quando você cria um volume, o Trident primeiro obtém o conjunto de pools de armazenamento para a classe de armazenamento desse volume e, se você especificar um protocolo para o volume, o Trident removerá esses pools de armazenamento que não podem fornecer o protocolo solicitado (por exemplo, um back-end do NetApp HCI/SolidFire não pode fornecer um volume baseado em arquivo enquanto um back-end do ONTAP nas não pode fornecer um volume baseado em bloco). O Trident aleatoriza a ordem desse conjunto resultante, para facilitar uma distribuição uniforme de volumes e, em seguida, iterar através dele, tentando provisionar o volume em cada pool de armazenamento por sua vez. Se for bem-sucedido em um, ele retorna com sucesso, registrando quaisquer falhas encontradas no processo. O Trident retorna uma falha **somente se** falhar em provisionar em **todos** os pools de armazenamento disponíveis para a classe de armazenamento e protocolo solicitados.

Criação de instantâneo de volume do Trident

Saiba mais sobre como o Trident lida com a criação de snapshots de volume para seus drivers.

Saiba mais sobre a criação de instantâneos de volume

- Para o `ontap-nas` , `ontap-san` , e `azure-netapp-files` Nos drivers, cada Volume Persistente (PV) é mapeado para um FlexVol volume. Como resultado, os snapshots de volume são criados como snapshots NetApp . A tecnologia de snapshots da NetApp oferece mais estabilidade, escalabilidade, capacidade de recuperação e desempenho do que as tecnologias de snapshots concorrentes. Essas cópias instantâneas são extremamente eficientes, tanto em termos de tempo necessário para criá-las quanto em espaço de armazenamento.
- Para `ontap-nas-flexgroup` o condutor, cada volume persistente (PV) é mapeado para um FlexGroup. Como resultado, os snapshots de volume são criados como snapshots do NetApp FlexGroup. A tecnologia Snapshot da NetApp oferece mais estabilidade, escalabilidade, capacidade de recuperação e desempenho do que as tecnologias de snapshot da concorrência. Essas cópias snapshot são extremamente eficientes no tempo necessário para criá-las e no espaço de storage.
- Para `ontap-san-economy` o driver, PVS mapeiam para LUNs criados em volumes FlexVol compartilhados VolumeOs snapshots de PVS são obtidos executando FlexClones do LUN associado. Com a tecnologia ONTAP FlexClone, é possível criar cópias dos maiores conjuntos de dados quase instantaneamente. As cópias compartilham blocos de dados com os pais, não consumindo storage, exceto o necessário para os metadados.
- Para `solidfire-san` o driver, cada PV mapeia para um LUN criado no cluster do software/NetApp HCI do NetApp Element. VolumeSnapshots são representados por instantâneos de elementos do LUN subjacente. Esses snapshots são cópias pontuais e ocupam apenas um pequeno espaço e recursos do sistema.
- Ao trabalhar com `ontap-nas` os drivers e `ontap-san`, os snapshots do ONTAP são cópias pontuais do FlexVol e consomem espaço no próprio FlexVol. Isso pode resultar na quantidade de espaço gravável no volume para reduzir com o tempo, à medida que os snapshots são criados/programados. Uma maneira

simples de lidar com isso é aumentar o volume redimensionando pelo Kubernetes. Outra opção é excluir snapshots que não são mais necessários. Quando um VolumeSnapshot criado pelo Kubernetes é excluído, o Trident excluirá o snapshot do ONTAP associado. Os snapshots do ONTAP que não foram criados pelo Kubernetes também podem ser excluídos.

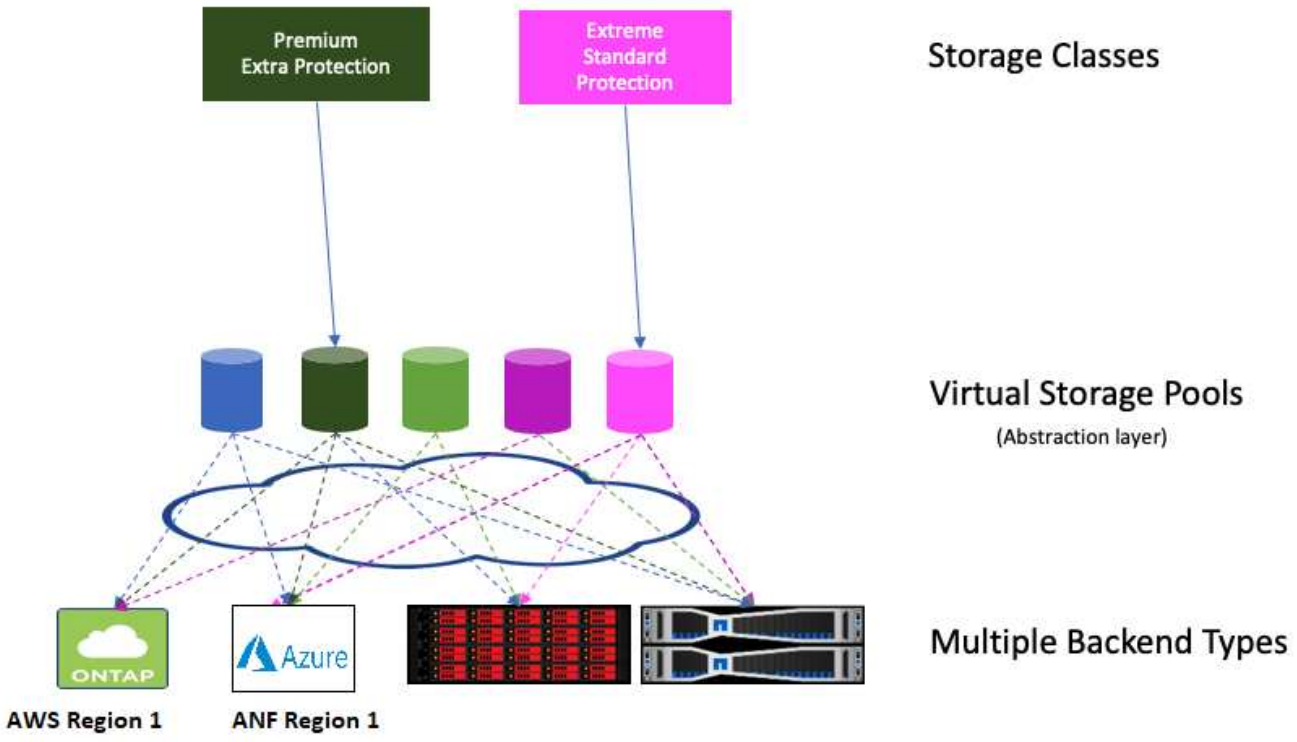
Com o Trident, você pode usar VolumeSnapshots para criar novos PVs a partir deles. A criação de PVs a partir desses snapshots é realizada utilizando a tecnologia FlexClone para backends ONTAP compatíveis. Ao criar um PV a partir de um snapshot, o volume de suporte é um FlexClone do volume pai do snapshot. O `solidfire-san` O driver utiliza clones de volume do software Element para criar PVs a partir de snapshots. Aqui, ele cria um clone a partir do instantâneo do elemento.

Configuração do pool de storage virtual do Trident

Os pools virtuais fornecem uma camada de abstração entre os backends de storage do Trident e o Kubernetes `StorageClasses`. Eles permitem que um administrador defina aspetos, como localização, desempenho e proteção para cada back-end de uma maneira comum e independente de back-end, sem `StorageClass` especificar qual backend físico, pool de back-end ou tipo de back-end usar para atender aos critérios desejados.

Saiba mais sobre pools virtuais

O administrador de armazenamento pode definir pools virtuais em qualquer um dos backends do Trident em um arquivo de definição JSON ou YAML.



Qualquer aspeto especificado fora da lista de pools virtuais é global para o back-end e se aplicará a todos os pools virtuais, enquanto cada pool virtual pode especificar um ou mais aspetos individualmente (substituindo

quaisquer aspetos globais de back-end).

- Ao definir pools virtuais, não tente reorganizar a ordem dos pools virtuais existentes em uma definição de back-end.
- Aconselhamos a não modificar atributos para um pool virtual existente. Você deve definir um novo pool virtual para fazer alterações.

A maioria dos aspetos são especificados em termos específicos de back-end. Fundamentalmente, os valores de aspeto não são expostos fora do driver do back-end e não estão disponíveis para correspondência em `StorageClasses`. em vez disso, o administrador define um ou mais rótulos para cada pool virtual. Cada rótulo é um par chave:valor, e os rótulos podem ser comuns em backends exclusivos. Assim como aspetos, os rótulos podem ser especificados por pool ou globais para o back-end. Ao contrário de aspetos, que têm nomes e valores predefinidos, o administrador tem total discrição para definir chaves de rótulo e valores conforme necessário. Por conveniência, os administradores de storage podem definir rótulos por pool virtual e volumes de grupo por rótulo.

Os rótulos do pool virtual podem ser definidos usando estes caracteres:

- letras maiúsculas `A-Z`
- letras minúsculas `a-z`
- números `0-9`
- sublinhados `_`
- hífens `-`

A `StorageClass` identifica qual pool virtual usar fazendo referência aos rótulos dentro de um parâmetro seletor. Os seletores de pool virtual suportam os seguintes operadores:

Operador	Exemplo	O valor do rótulo de um pool deve:
`=`	desempenho superior	Correspondência
`!=`	performance! extrema	Não corresponde
`in`	localização em (leste, oeste)	Esteja no conjunto de valores
`notin`	notificação de desempenho (prata, bronze)	Não estar no conjunto de valores
`<key>`	proteção	Existe com qualquer valor
`!<key>`	!proteção	Não existe

Gerenciamento de grupo de acesso ao volume do Trident

Saiba mais sobre como o Trident usa "grupos de acesso de volume".

Ignore esta seção se você estiver usando CHAP, que é recomendado para simplificar o gerenciamento e evitar o limite de escala descrito abaixo. Além disso, se você estiver usando o Trident no modo CSI, você pode ignorar esta seção. O Trident usa o CHAP quando instalado como um provisionador de CSI aprimorado.

Saiba mais sobre grupos de acesso de volume

O Trident pode usar grupos de acesso de volume para controlar o acesso aos volumes provisionados. Se o CHAP estiver desativado, ele espera encontrar um grupo de acesso chamado `trident`, a menos que você especifique um ou mais IDs de grupo de acesso na configuração.

Embora o Trident associe novos volumes aos grupos de acesso configurados, ele não cria nem gerencia os próprios grupos de acesso. Os grupos de acesso devem existir antes que o back-end de storage seja adicionado ao Trident e precisam conter as IQNs iSCSI de todos os nós do cluster do Kubernetes que poderiam potencialmente montar os volumes provisionados por esse back-end. Na maioria das instalações, isso inclui cada nó de trabalho no cluster.

Para clusters de Kubernetes com mais de 64 nós, você deve usar vários grupos de acesso. Cada grupo de acesso pode conter até 64 IQNs e cada volume pode pertencer a quatro grupos de acesso. Com o máximo de quatro grupos de acesso configurados, qualquer nó em um cluster de até 256 nós de tamanho poderá acessar qualquer volume. Para obter os limites mais recentes dos grupos de acesso de volume, "aqui" consulte a .

Se você estiver modificando a configuração de uma que esteja usando o grupo de acesso padrão `trident` para outra que também use outras, inclua a ID do `trident` grupo de acesso na lista.

Início rápido para Trident

Você pode instalar o Trident e começar a gerenciar recursos de storage em algumas etapas. Antes de começar, reveja "Requisitos da Trident"o .

Para Docker, "Trident para Docker"consulte .



Prepare o nó de trabalho

Todos os nós de trabalho no cluster do Kubernetes precisam ser capazes de montar os volumes provisionados para os pods.

"Prepare o nó de trabalho"



Instale o Trident

O Trident oferece vários métodos e modos de instalação otimizados para uma variedade de ambientes e organizações.

"Instale o Trident"



Crie um backend

Um back-end define a relação entre o Trident e um sistema de storage. Ele informa à Trident como se comunicar com esse sistema de storage e como o Trident deve provisionar volumes a partir dele.

"Configurar um back-end" para o seu sistema de storage



Crie um Kubernetes StorageClass

O objeto Kubernetes StorageClass especifica o Trident como o provisionador e permite que você crie uma classe de storage para provisionar volumes com atributos personalizáveis. O Trident cria uma classe de storage correspondente para objetos Kubernetes que especificam o provisionador do Trident.

"Crie uma classe de armazenamento"

5 **Provisionar um volume**

Um *Persistentvolume* (PV) é um recurso de armazenamento físico provisionado pelo administrador de cluster em um cluster do Kubernetes. O *PersistentVolumeClaim* (PVC) é um pedido de acesso ao Persistentvolume no cluster.

Crie um Persistentvolume (PV) e um PersistentVolumeClaim (PVC) que use o Kubernetes StorageClass configurado para solicitar acesso ao PV. Em seguida, pode montar o PV num pod.

"Provisionar um volume"

O que se segue?

Agora você pode adicionar backends adicionais, gerenciar classes de armazenamento, gerenciar backends e executar operações de volume.

Requisitos da Trident

Antes de instalar o Trident, você deve rever estes requisitos gerais do sistema. Backends específicos podem ter requisitos adicionais.

Informações críticas sobre o Trident

Você deve ler as seguintes informações críticas sobre o Trident.

**▼ informações críticas sobre o Trident **

- O Trident 26.10 é compatível com o Kubernetes das versões 1.27 a 1.37. Consulte a lista de imagens específicas para cada versão abaixo antes de atualizar o Kubernetes.
- O Trident reforça estritamente o uso de configuração multipathing em ambientes SAN, com um valor recomendado de `find_multipaths: no` no arquivo multipath.conf.

  O uso de configuração não multipathing ou o uso `find_multipaths: yes` de ou `find_multipaths: smart` valor no arquivo multipath.conf resultará em falhas de montagem. A Trident recomenda o uso de `find_multipaths: no` desde a versão 21,07.

Frontens suportados (orquestradores)

O Trident é compatível com vários mecanismos de contêiner e orquestradores, incluindo os seguintes:

- Anthos On-Prem (VMware) e Anthos em bare metal 1,16
- Kubernetes 1.27 - 1.37
- OpenShift 4.12, 4.14 - 4.22 (se você planeja usar a preparação de nó iSCSI com OpenShift 4.19, a versão

mínima do Trident suportada é 25.06.1.)

> A Trident continua a oferecer suporte a versões mais antigas do OpenShift em alinhamento com o"Ciclo de vida da versão do Red Hat Extended Update Support (EUS)" , mesmo que dependam de versões do Kubernetes que não são mais oficialmente suportadas pelo upstream. Ao instalar o Trident nesses casos, você pode ignorar com segurança quaisquer mensagens de aviso sobre a versão do Kubernetes.

- Rancher Kubernetes Engine 2 (RKE2) v1.28.x - 1.37.x
- Talos OS para NFSv4, iSCSI e NVMe/TCP. As seguintes extensões do sistema Talos são necessárias para cada protocolo; confirme esses requisitos com a Sidero Labs.
  - NFSv4: `siderolabs/nfs-utils`
  - iSCSI: `siderolabs/iscsi-tools`, `siderolabs/multipath-tools`, e `siderolabs/trident-iscsi-tools`
  - NVMe/TCP: `siderolabs/nvme-cli`, `siderolabs/util-linux-tools`, e `siderolabs/trident-iscsi-tools`

O Trident também trabalha com uma série de outras ofertas do Kubernetes totalmente gerenciadas e autogeridas, incluindo o Google Kubernetes Engine (GKE), o Amazon Elastic Kubernetes Services (EKS), o

O Trident e o ONTAP podem ser usados como um provedor de storage para "KubeVirt"o .

> Antes de atualizar um cluster do Kubernetes do 1,25 para o 1,26 ou posterior que tenha o Trident instalado, "Atualize uma instalação do Helm"consulte a .

Backends suportados (armazenamento)

Para usar o Trident, você precisa de um ou mais dos seguintes backends suportados:

- Amazon FSX para NetApp ONTAP
- Azure NetApp Files
- Cloud Volumes ONTAP
- Google Cloud NetApp volumes
- NetApp All SAN Array (ASA)
- Sistemas FAS, AFF ou ASA r2 locais (iSCSI, NVMe/TCP e FC) executando versões do ONTAP com suporte total ou limitado da NetApp . Ver "Suporte à versão de software".
- Software NetApp HCI/Element 11 ou superior

Suporte ao Trident para virtualização KubeVirt e OpenShift

Drivers de armazenamento suportados:

O Trident suporta os seguintes drivers ONTAP para virtualização KubeVirt e OpenShift:

- ONTAP-nas
- ONTAP-san (iSCSI, FCP, NVMe em TCP)
- ONTAP-San-Economy (apenas iSCSI)

Pontos a considerar:

- Atualize a classe de armazenamento para ter o `fsType` parâmetro (por exemplo: `fsType: "ext4"`) No ambiente de virtualização OpenShift. Se necessário, defina o modo de volume para bloquear explicitamente usando o `volumeMode=Block` parâmetro no `dataVolumeTemplates` para notificar CDI para criar volumes de dados de bloco.
- *RWX modo de acesso para drivers de armazenamento em bloco*: Os drivers ONTAP-San (iSCSI, NVMe/TCP, FC) e ONTAP-san-Economy (iSCSI) são suportados apenas com "VolumeMode: Block" (dispositivo bruto). Para esses drivers, o `fstype` parâmetro não pode ser usado porque os volumes são fornecidos no modo de dispositivo bruto.
- Para fluxos de trabalho de migração em tempo real em que o modo de acesso RWX é necessário, essas combinações são suportadas:
  - DE NFS `volumeMode=Filesystem`
  - ISCSI `volumeMode=Block` (dispositivo bruto)
  - NVMe/TCP `volumeMode=Block` (dispositivo bruto)
  - FC `volumeMode=Block` (dispositivo bruto)

Suporte do Trident para Kubernetes em Kubernetes e OpenShift planos de controle hospedados

O Trident qualifica totalmente a topologia Kubernetes on Kubernetes. Clusters Kubernetes cujos nós são executados como máquinas virtuais KubeVirt em um cluster de infraestrutura (host). Essa topologia sustenta ofertas de plano de controle hospedado, como planos de controle hospedados (HCP) do OpenShift na virtualização do OpenShift.

> Essa configuração funcionava em versões anteriores, mas a qualificação completa começou com a versão 26.06.

O cluster convidado (inquilino) utiliza o driver CSI KubeVirt (`kubevirt-csi` como seu provedor de armazenamento. Quando uma carga de trabalho no cluster convidado solicita um `PersistentVolumeClaim`, o driver CSI KubeVirt cria um volume correspondente no cluster de infraestrutura. O Trident provisiona o volume a partir do ONTAP, e o volume é conectado dinamicamente à máquina virtual do nó convidado. O Trident é executado somente no cluster de infraestrutura. O cluster convidado consome o armazenamento do ONTAP indiretamente por meio do driver CSI KubeVirt.

Requisitos:

- Trident está instalado no cluster de infraestrutura com um backend ONTAP compatível.
- Existe uma classe de armazenamento Trident no cluster de infraestrutura para que o driver CSI KubeVirt mapeie os volumes do cluster convidado.
- KubeVirt ou OpenShift Virtualization é implementado no cluster de infraestrutura, e o driver CSI do KubeVirt é instalado em cada cluster convidado.

Componente	Versão
Trident	26,10
Kubernetes (infraestrutura e cluster convidado)	1,37
KubeVirt	1.9.0
Importador de dados em contêineres (CDI)	1,65

Pontos a considerar:

- No cluster convidado, os volumes aparecem como dispositivos de bloco hot-plugged nas máquinas virtuais dos nós convidados.
- O fluxo de trabalho do driver CSI KubeVirt não requer Trident no cluster convidado. Você configura o backend ONTAP e a classe de armazenamento no cluster de infraestrutura.

Instale Trident diretamente no cluster convidado:

Você também pode instalar Trident diretamente no cluster convidado (tenant). Instale Trident no cluster convidado quando precisar do conjunto completo de recursos do Trident, como snapshots, clones, redimensionamento de volumes e importação, que `kubevirt-csi` não são fornecidos por si só.

No cluster convidado, utilize protocolos baseados em IP: NFS, iSCSI e NVMe/TCP. O cluster convidado não suporta Fibre Channel (FC) porque os nós convidados são máquinas virtuais sem conectividade FC direta.

Requisitos de recursos

A tabela abaixo resume os recursos disponíveis com esta versão do Trident e as versões do Kubernetes compatíveis.

Recurso	Versão do Kubernetes	É necessário ter portões?
Trident	1.27 - 1.37	Não
Instantâneos de volume	1.27 - 1.37	Não
PVC a partir de instantâneos de volume	1.27 - 1.37	Não
Redimensionamento iSCSI PV	1.27 - 1.37	Não
ONTAP bidirectional CHAP	1.27 - 1.37	Não
Políticas de exportação dinâmica	1.27 - 1.37	Não
Operador Trident	1.27 - 1.37	Não
Topologia de CSI	1.27 - 1.37	Não

Sistemas operacionais de host testados

Embora o Trident não suporte oficialmente sistemas operacionais específicos, sabe-se que os seguintes procedimentos funcionam:

- Versões do Red Hat Enterprise Linux CoreOS (RHCOS) suportadas pela plataforma OpenShift Container em AMD64 e ARM64
- Red Hat Enterprise Linux (RHEL) 8 ou posterior em AMD64 e ARM64

> O NVMe/TCP requer o RHEL 9 ou posterior.

- Ubuntu 22.04 LTS ou posterior em AMD64 e ARM64
- Windows Server 2022
- SUSE Linux Enterprise Server (SLES) 15 ou posterior

Por padrão, o Trident é executado em um contentor e, portanto, será executado em qualquer trabalhador

Linux. No entanto, esses funcionários precisam ser capazes de montar os volumes que o Trident fornece usando o cliente NFS padrão ou iniciador iSCSI, dependendo dos backends que você está usando.

O `tridentctl` utilitário também é executado em qualquer uma dessas distribuições do Linux.

Configuração de host

Todos os nós de trabalho no cluster do Kubernetes precisam ser capazes de montar os volumes provisionados para os pods. Para preparar os nós de trabalho, é necessário instalar ferramentas NFS, iSCSI ou NVMe com base na seleção de driver.

"Prepare o nó de trabalho"

Configuração do sistema de storage

O Trident pode exigir alterações em um sistema de storage antes que uma configuração de back-end possa usá-lo.

"Configurar backends"

Portas Trident

O Trident requer acesso a portas específicas para comunicação.

"Portas Trident"

Imagens de contêineres e versões correspondentes do Kubernetes

Para instalações com ar-gapped, a lista a seguir é uma referência das imagens de contentor necessárias para instalar o Trident. Use o `tridentctl images` comando para verificar a lista de imagens de contentor necessárias.

A lista de imagens do Trident 26.10 abaixo inclui as imagens para o Kubernetes 1.37.

Imagens de contêiner necessárias para o Trident 26.10.1

Versões do Kubernetes	Imagem do recipiente
v1.27.0, v1.28.0, v1.29.0, v1.30.0, v1.31.0, v1.32.0, v1.33.0, v1.34.0, v1.35.0, v1.36.0, v1.37.0	• docker.io/netapp/trident:26.10.1 • docker.io/netapp/trident-autosupport:26.10 • docker.io/netapp/trident-operator:26.10.1 (opcional) • registry.k8s.io/sig-storage/csi-attacher:v4.11.0 • registry.k8s.io/sig-storage/csi-node-driver-registrar:v2.17.0 • registry.k8s.io/sig-storage/csi-provisioner:v6.3.0 • registry.k8s.io/sig-storage/csi-resizer:v2.1.0 • registry.k8s.io/sig-storage/csi-snapshotter:v8.6.0

Imagens de contêiner necessárias para o Trident 26.06.2

Versões do Kubernetes	Imagem do recipiente
v1.27.0, v1.28.0, v1.29.0, v1.30.0, v1.31.0, v1.32.0, v1.33.0, v1.34.0, v1.35.0, v1.36.0	• docker.io/netapp/trident:26.06.2 • docker.io/netapp/trident-autosupport:26.06 • docker.io/netapp/trident-operator:26.06.2 (opcional) • registry.k8s.io/sig-storage/csi-attacher:v4.11.0 • registry.k8s.io/sig-storage/csi-node-driver-registrar:v2.18.0 • registry.k8s.io/sig-storage/csi-provisioner:v6.3.0 • registry.k8s.io/sig-storage/csi-resizer:v2.1.0 • registry.k8s.io/sig-storage/csi-snapshotter:v8.6.0

Instale o Trident

Saiba mais sobre a instalação do Trident

Para garantir que o Trident possa ser instalado em uma ampla variedade de ambientes e organizações, o NetApp oferece várias opções de instalação. Você pode instalar o Trident usando o operador Trident (manualmente ou usando o Helm) ou com `tridentctl`o . Este tópico fornece informações importantes para selecionar o processo de instalação certo para você.

Informações críticas sobre Trident 26.10

Você deve ler as seguintes informações críticas sobre o Trident.

▼ ** informações críticas sobre o Trident **

- Kubernetes 1.37 agora é compatível com Trident. Atualize Trident antes de atualizar o Kubernetes.
- O Trident reforça estritamente o uso de configuração multipathing em ambientes SAN, com um valor recomendado de `find_multipaths: no` no arquivo multipath.conf.

  O uso de configuração não multipathing ou o uso `find_multipaths: yes` de ou `find_multipaths: smart` valor no arquivo multipath.conf resultará em falhas de montagem. A Trident recomenda o uso de `find_multipaths: no` desde a versão 21,07.

Antes de começar

Independentemente do seu caminho de instalação, você deve ter:

- Privilégios completos em um cluster Kubernetes compatível, executando uma versão compatível do Kubernetes e com os requisitos de recursos ativados. Revise a "requisitos" para obter detalhes.
- Acesso a um sistema de storage NetApp compatível.
- Funcionalidade de montar volumes de todos os nós de trabalho do Kubernetes.
- Um host Linux com `kubectl` (ou `oc`, se você estiver usando o OpenShift) instalado e configurado para gerenciar o cluster do Kubernetes que deseja usar.
- A `KUBECONFIG` variável de ambiente configurada para apontar para a configuração do cluster do Kubernetes.
- Se você estiver usando o Kubernetes com Docker Enterprise "Siga os passos para ativar o acesso CLI", .
- O cluster precisa dar suporte a workloads privilegiados.

Se você não se familiarizou com o "conceitos básicos", agora é um grande momento para fazer isso.

Escolha o método de instalação

Selecione o método de instalação mais adequado para você. Você também deve analisar as considerações para "movendo-se entre métodos" antes de tomar sua decisão.

Utilizando o operador Trident

Seja implantando manualmente ou usando o Helm, o operador Trident é uma ótima maneira de simplificar a instalação e gerenciar dinamicamente os recursos do Trident. Você pode até "Personalize a implantação do seu operador Trident" usando os atributos no `TridentOrchestrator` recurso personalizado (CR).

Os benefícios de usar o operador Trident incluem:

▼ **Criação de objeto Trident **

O operador Trident cria automaticamente os seguintes objetos para a versão do Kubernetes.

- ServiceAccount para o operador
- ClusterRole e ClusterRoleBinding para o ServiceAccount
- PodSecurityPolicy dedicada (para Kubernetes 1,25 e versões anteriores)
- O próprio operador

▼ **Accountability de recursos**

O operador Trident com escopo de cluster gerencia os recursos associados a uma instalação do Trident no nível do cluster. Isso atenua erros que podem ser causados ao manter recursos com escopo de cluster usando um operador com escopo de namespace. Isso é essencial para a auto-cura e correção.

▼ ** capacidade de autorrecuperação **

O operador monitora a instalação do Trident e toma ativamente medidas para resolver problemas, como quando a implantação é excluída ou se for modificada acidentalmente. É criado um `trident-operator-<generated-id>` pod que associa um `TridentOrchestrator` CR a uma instalação do Trident. Isso garante que haja apenas uma instância do Trident no cluster e controla sua configuração, garantindo que a instalação seja idempotente. Quando as alterações são feitas na instalação (como, por exemplo, a exclusão do daemonset de implantação ou nó), o operador as identifica e as corrige individualmente.

▼ ** atualizações fáceis para existente**

Você pode facilmente atualizar uma implantação existente com o operador. Você só precisa editar o `TridentOrchestrator` CR para fazer atualizações em uma instalação.

Por exemplo, considere um cenário em que você precisa habilitar o Trident para gerar logs de depuração. Para fazer isso, corrija o `TridentOrchestrator` para definir `spec.debug` como `true`:

```
kubectl patch torc <trident-orchestrator-name> -n trident --type=merge
-p '{"spec":{"debug":true}}'
```

Após `TridentOrchestrator` a atualização, o operador processa as atualizações e corrige a instalação existente. Isso pode acionar a criação de novos pods para modificar a instalação de acordo.

▼ ** Clean restablation**

O operador Trident com escopo de cluster permite a remoção limpa de recursos com escopo de cluster. Os usuários podem desinstalar completamente o Trident e reinstalar facilmente.

▼ ** handling de atualização automática do Kubernetes**

Quando a versão do Kubernetes do cluster é atualizada para uma versão compatível, a operadora atualiza uma instalação existente do Trident automaticamente e a altera para garantir que ela atenda aos requisitos da versão do Kubernetes.

Se o cluster for atualizado para uma versão não suportada, o operador impede a instalação do Trident. Se o Trident já tiver sido instalado com o operador, um aviso será exibido para indicar que o Trident está instalado em uma versão do Kubernetes não suportada.

Utilização `tridentctl`

Se você tiver uma implantação existente que deve ser atualizada ou se você estiver procurando personalizar altamente sua implantação, considere o . Este é o método convencional de implantação do Trident.

Você pode gerar os manifestos para recursos do Trident. Isso inclui a implantação, o daemonset, a conta de serviço e a função de cluster que o Trident cria como parte de sua instalação.

A partir da versão 22,04, as chaves AES não serão mais regeneradas sempre que o Trident for instalado. Com esta versão, o Trident instalará um novo objeto secreto que persiste entre as instalações. Isso significa que `tridentctl` no 22,04 pode desinstalar versões anteriores do Trident, mas versões anteriores não podem desinstalar instalações do 22,04. Selecione a instalação apropriada *method*.

Escolha o modo de instalação

Determine seu processo de implantação com base no *modo de instalação* (padrão, Offline ou remoto) exigido pela sua organização.

Instalação padrão

Esta é a maneira mais fácil de instalar o Trident e funciona para a maioria dos ambientes que não impõem restrições de rede. O modo de instalação padrão usa Registros padrão para armazenar (`registry.k8s.io`imagens Trident (`docker.io`) e CSI necessárias.

Quando você usa o modo padrão, o instalador do Trident:

- Obtém as imagens de contentor pela Internet
- Cria um daemonset de implantação ou nó, que gira pods do Trident em todos os nós elegíveis no cluster do Kubernetes

Instalação offline

O modo de instalação off-line pode ser necessário em um local seguro ou protegido. Nesse cenário, você pode criar um único Registro privado espelhado ou dois Registros espelhados para armazenar imagens Trident e CSI necessárias.

Independentemente da configuração do Registro, as imagens CSI devem residir em um Registro.

Instalação remota

Aqui está uma visão geral de alto nível do processo de instalação remota:

- Implante a versão apropriada do `kubectl` na máquina remota de onde você deseja implantar o Trident.
- Copie os arquivos de configuração do cluster do Kubernetes e defina a `KUBECONFIG` variável de ambiente na máquina remota.

- Inicie um `kubectl get nodes` comando para verificar se você pode se conetar ao cluster do Kubernetes necessário.
- Conclua a implementação a partir da máquina remota utilizando as etapas de instalação padrão.

Selecione o processo com base no seu método e modo

Depois de tomar suas decisões, selecione o processo apropriado.

Método	Modo de instalação
Operador Trident (manualmente)	"Instalação padrão" "Instalação offline"
Operador Trident (Helm)	"Instalação padrão" "Instalação offline"
`tridentctl`	"Instalação padrão ou offline"

Movendo-se entre os métodos de instalação

Você pode decidir alterar seu método de instalação. Antes de fazer isso, considere o seguinte:

- Sempre use o mesmo método para instalar e desinstalar o Trident. Se você tiver implantado com `tridentctl`o , você deve usar a versão apropriada `tridentctl` do binário para desinstalar o Trident. Da mesma forma, se você estiver implantando com o operador, edite o `TridentOrchestrator` CR e defina `spec.uninstall=true` como desinstalar o Trident.
- Se você tiver uma implantação baseada no operador que deseja remover e usar `tridentctl` para implantar o Trident, primeiro edite `TridentOrchestrator` e defina `spec.uninstall=true` como desinstalar o Trident. Em seguida, exclua `TridentOrchestrator` e a implantação do operador. Você pode instalar usando `tridentctl`o .
- Se você tiver uma implantação manual baseada em operador e quiser usar a implantação do operador Trident baseada em Helm, primeiro desinstale manualmente o operador e depois execute a instalação do Helm. Isso permite que o Helm implante o operador Trident com os rótulos e anotações necessários. Se você não fizer isso, a implantação do operador Trident baseado em Helm falhará com erro de validação de rótulo e erro de validação de anotação.
- Se você tem um `tridentctl` implantação baseada em Helm ou em Operador sem desinstalar o Trident.

Outras opções de configuração conhecidas

Ao instalar o Trident em produtos do portfólio VMware Tanzu:

- A `--kubelet-dir` bandeira deve ser definida para a localização do diretório kubelet. Por padrão, isso é `/var/vcap/data/kubelet`.

  Especificar a localização do kubelet usando `--kubelet-dir` é conhecido por funcionar para o Operador Trident, Helm e `tridentctl` implantações.

Instale usando o operador Trident

Implantar manualmente o operador Trident (modo padrão)

Você pode implantar manualmente o operador Trident para instalar o Trident. Este processo aplica-se a instalações onde as imagens de contentor exigidas pelo Trident não são armazenadas em um Registro privado. Se tiver um registo de imagens privado, utilize o "processo para implantação off-line".

Informações críticas sobre Trident 26.10

Você deve ler as seguintes informações críticas sobre o Trident.

▼ informações críticas sobre o Trident

- O Trident 26.10 oferece suporte ao Kubernetes 1.27 a 1.37. Consulte a lista de imagens específicas da versão em "Requisitos da Trident" antes de atualizar o Kubernetes.
- O Trident reforça estritamente o uso de configuração multipathing em ambientes SAN, com um valor recomendado de `find_multipaths: no` no arquivo multipath.conf.

  O uso de configuração não multipathing ou o uso `find_multipaths: yes` de ou `find_multipaths: smart` valor no arquivo multipath.conf resultará em falhas de montagem. A Trident recomenda o uso de `find_multipaths: no` desde a versão 21,07.

Implante manualmente o operador Trident e instale o Trident

Revise "a visão geral da instalação" para garantir que você atendeu aos pré-requisitos de instalação e selecionou a opção de instalação correta para o seu ambiente.

Antes de começar

Antes de iniciar a instalação, faça login no host Linux e verifique se ele está gerenciando um sistema operacional em funcionamento "Cluster compatível com Kubernetes" e se você possui os privilégios necessários.

Com OpenShift, use `oc` em vez de `kubectl` em todos os exemplos que se seguem, e faça login como **system:admin** primeiro executando `oc login -u system:admin` ou `oc login -u kube-admin`.

▼ Details

1. Verifique sua versão do Kubernetes:

```
kubectl version
```

2. Verifique o Privileges do administrador do cluster:

```
kubectl auth can-i '*' '*' --all-namespaces
```

3. Verifique se você pode iniciar um pod que usa uma imagem do Docker Hub e alcançar seu sistema de armazenamento pela rede de pod:

```
kubectl run -i --tty ping --image=busybox --restart=Never --rm -- \
  ping <management IP>
```

Passo 1: Baixe o pacote de instalação do Trident

O pacote de instalação do Trident contém tudo o que você precisa para implantar o operador Trident e instalar o Trident. Baixe e extraia a versão mais recente do instalador do Trident a partir "A seção *assets* no GitHub"do .

```
wget https://github.com/NetApp/trident/releases/download/v26.10.0/trident-
installer-26.10.1.tar.gz
tar -xf trident-installer-26.10.1.tar.gz
cd trident-installer
```

Passo 2: Crie o `TridentOrchestrator` CRD

Crie o `TridentOrchestrator` Definição de Recurso Personalizado (CRD). Você criará um `TridentOrchestrator` Recurso personalizado mais tarde. Use a versão CRD YAML apropriada em `deploy/crds` para criar o `TridentOrchestrator` CRD.

```
kubectl create -f
deploy/crds/trident.netapp.io_tridentorchestrators_crd_post1.16.yaml
```

Etapa 3: Implante o operador Trident

O instalador do Trident fornece um arquivo de pacote que pode ser usado para instalar o operador e criar objetos associados. O arquivo bundle é uma maneira fácil de implantar o operador e instalar o Trident usando uma configuração padrão.

- Para clusters que executam o Kubernetes 1,24, `bundle_pre_1_25.yaml` use o .
- Para clusters que executam o Kubernetes 1,25 ou posterior, `bundle_post_1_25.yaml` use o .

Antes de começar

- Por padrão, o instalador do Trident implanta o operador no `trident` namespace. Se o `trident` namespace não existir, crie-o usando:

```
kubectl apply -f deploy/namespace.yaml
```

- Para implantar o operador em um namespace diferente do `trident` namespace, atualizar `serviceaccount.yaml` `clusterrolebinding.yaml` `operator.yaml` e gerar o arquivo do pacote usando o `kustomization.yaml`.

a. Crie o `kustomization.yaml` usando o seguinte comando onde *<bundle.yaml>* está `bundle_pre_1_25.yaml` ou `bundle_post_1_25.yaml` baseado na sua versão do Kubernetes.

```
cp deploy/kustomization_<bundle.yaml> deploy/kustomization.yaml
```

b. Compile o pacote usando o seguinte comando onde *<bundle.yaml>* está `bundle_pre_1_25.yaml` ou `bundle_post_1_25.yaml` baseado na sua versão do Kubernetes.

```
kubectl kustomize deploy/ > deploy/<bundle.yaml>
```

Passos

1. Crie os recursos e implante o operador:

```
kubectl create -f deploy/<bundle.yaml>
```

2. Verifique se o operador, a implantação e as replicasets foram criados.

```
kubectl get all -n <operator-namespace>
```

Deve haver apenas **uma instância** do operador em um cluster do Kubernetes. Não crie várias implantações do operador Trident.

Passo 4: Crie o `TridentOrchestrator` e instale o Trident

Agora você pode criar e instalar o `TridentOrchestrator` Trident. Opcionalmente, você pode "Personalize a instalação do Trident" usar os atributos na `TridentOrchestrator` especificação.

```
kubectl create -f deploy/crds/tridentorchestrator_cr.yaml
tridentorchestrator.trident.netapp.io/trident created

kubectl describe torc trident

Name:         trident
Namespace:
Labels:       <none>
Annotations:  <none>
API Version:  trident.netapp.io/v1
Kind:         TridentOrchestrator
...
Spec:
  Debug:      true
  Namespace:  trident
```

```
  nodePrep:
  - iscsi
Status:
  Current Installation Params:
    IPv6:                      false
    Autosupport Hostname:
    Autosupport Image:         netapp/trident-autosupport:26.10
    Autosupport Proxy:
    Autosupport Serial Number:
    Debug:                     true
    Image Pull Secrets:
    Image Registry:
    k8sTimeout:            30
    Kubelet Dir:           /var/lib/kubelet
    Log Format:            text
    Silence Autosupport:   false
    Trident Image:         netapp/trident:26.10.1
  Message:                   Trident installed  Namespace:
trident
  Status:                    Installed
  Version:                   v26.10.0
Events:
    Type Reason Age From Message ---- ------ ---- ---- -------Normal
    Installing 74s trident-operator.netapp.io Installing Trident Normal
    Installed 67s trident-operator.netapp.io Trident installed
```

Verifique a instalação

Existem várias maneiras de verificar sua instalação.

`TridentOrchestrator`Usando o status

O status de `TridentOrchestrator` indica se a instalação foi bem-sucedida e exibe a versão do Trident instalado. Durante a instalação, o status das `TridentOrchestrator` alterações de `Installing` para `Installed`. Se você observar o `Failed` status e o operador não conseguir recuperar sozinho, "verifique os logs".

Estado	Descrição
A instalar	O operador está a instalar o Trident utilizando este `TridentOrchestrator` CR.
Instalado	O Trident foi instalado com sucesso.
Desinstalação	O operador está desinstalando o Trident, porque `spec.uninstall=true`.
Desinstalado	O Trident é desinstalado.

Estado	Descrição
Falha	O operador não pôde instalar, corrigir, atualizar ou desinstalar o Trident; o operador tentará recuperar automaticamente deste estado. Se este estado persistir, será necessário resolver o problema.
A atualizar	O operador está atualizando uma instalação existente.
Erro	O `TridentOrchestrator` não é utilizado. Outro já existe.

Usando o status de criação do pod

Você pode confirmar se a instalação do Trident foi concluída revisando o status dos pods criados:

```
kubectl get pods -n trident

NAME                                READY   STATUS    RESTARTS
AGE
trident-controller-7d466bf5c7-v4cpw        6/6     Running   0
1m
trident-node-linux-mr6zc                   2/2     Running   0
1m
trident-node-linux-xrp7w                   2/2     Running   0
1m
trident-node-linux-zh2jt                   2/2     Running   0
1m
trident-operator-766f7b8658-ldzsv          1/1     Running   0
3m
```

Utilização `tridentctl`

Você pode usar `tridentctl` para verificar a versão do Trident instalada.

```
./tridentctl -n trident version

+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 26.10.1        | 26.10.1        |
+----------------+----------------+
```

Implantar manualmente o operador Trident (modo off-line)

Você pode implantar manualmente o operador Trident para instalar o Trident. Este

processo aplica-se a instalações onde as imagens de contentor exigidas pelo Trident são armazenadas em um Registro privado. Se não tiver um registo de imagens privado, utilize o "processo para implantação padrão".

Informações críticas sobre o Trident

Você deve ler as seguintes informações críticas sobre o Trident.

▼ ** informações críticas sobre o Trident **

- O Trident 26.10 oferece suporte ao Kubernetes 1.27 a 1.37. Consulte a lista de imagens específicas da versão em "Requisitos da Trident" antes de atualizar o Kubernetes.
- O Trident reforça estritamente o uso de configuração multipathing em ambientes SAN, com um valor recomendado de `find_multipaths: no` no arquivo multipath.conf.

  O uso de configuração não multipathing ou o uso `find_multipaths: yes` de ou `find_multipaths: smart` valor no arquivo multipath.conf resultará em falhas de montagem. A Trident recomenda o uso de `find_multipaths: no` desde a versão 21,07.

Implante manualmente o operador Trident e instale o Trident

Revise "a visão geral da instalação" para garantir que você atendeu aos pré-requisitos de instalação e selecionou a opção de instalação correta para o seu ambiente.

Antes de começar

Faça login no host Linux e verifique se ele está gerenciando um sistema operacional em funcionamento "Cluster compatível com Kubernetes" e se você possui os privilégios necessários.

Com OpenShift, use `oc` em vez de `kubectl` em todos os exemplos que se seguem, e faça login como **system:admin** primeiro executando `oc login -u system:admin` ou `oc login -u kube-admin`.

▼ **Details**

1. Verifique sua versão do Kubernetes:

```
kubectl version
```

2. Verifique o Privileges do administrador do cluster:

```
kubectl auth can-i '*' '*' --all-namespaces
```

3. Verifique se você pode iniciar um pod que usa uma imagem do Docker Hub e alcançar seu sistema de armazenamento pela rede de pod:

```
kubectl run -i --tty ping --image=busybox --restart=Never --rm -- \
```

```
ping <management IP>
```

Passo 1: Baixe o pacote de instalação do Trident

O pacote de instalação do Trident contém tudo o que você precisa para implantar o operador Trident e instalar o Trident. Baixe e extraia a versão mais recente do instalador do Trident a partir "A seção *assets* no GitHub"do .

```
wget https://github.com/NetApp/trident/releases/download/v26.10.0/trident-installer-26.10.1.tar.gz
tar -xf trident-installer-26.10.1.tar.gz
cd trident-installer
```

Passo 2: Crie o `TridentOrchestrator` CRD

Crie o `TridentOrchestrator` Definição de Recurso Personalizado (CRD). Você criará um `TridentOrchestrator` Recursos personalizados mais tarde. Use a versão CRD YAML apropriada em `deploy/crds` para criar o `TridentOrchestrator` CRD:

```
kubectl create -f deploy/crds/<VERSION>.yaml
```

Etapa 3: Atualize a localização do Registro no operador

Em `/deploy/operator.yaml`, atualize `image: docker.io/netapp/trident-operator:26.10.1` para refletir a localização do seu registro de imagens. Seu "Imagens de Trident e CSI" pode estar localizado em um registro ou em registros diferentes, mas todas as imagens CSI devem estar no mesmo registro. Por exemplo:

- `image: <your-registry>/trident-operator:26.10.1` se todas as suas imagens estiverem localizadas em um único registro.
- `image: <your-registry>/netapp/trident-operator:26.10.1` se a sua imagem Trident estiver localizada em um registro diferente das suas imagens CSI.

Etapa 4: Implante o operador Trident

O instalador do Trident fornece um arquivo de pacote que pode ser usado para instalar o operador e criar objetos associados. O arquivo bundle é uma maneira fácil de implantar o operador e instalar o Trident usando uma configuração padrão.

- Para clusters que executam o Kubernetes 1,24, `bundle_pre_1_25.yaml` use o .
- Para clusters que executam o Kubernetes 1,25 ou posterior, `bundle_post_1_25.yaml` use o .

Antes de começar

- Por padrão, o instalador do Trident implanta o operador no `trident` namespace. Se o `trident` namespace não existir, crie-o usando:

```
kubectl apply -f deploy/namespace.yaml
```

- Para implantar o operador em um namespace diferente do `trident` namespace, atualizar `serviceaccount.yaml` `clusterrolebinding.yaml` `operator.yaml` e gerar o arquivo do pacote usando o `kustomization.yaml`.

  a. Crie o `kustomization.yaml` usando o seguinte comando onde *<bundle.yaml>* está `bundle_pre_1_25.yaml` ou `bundle_post_1_25.yaml` baseado na sua versão do Kubernetes.

  ```
  cp deploy/kustomization_<bundle.yaml> deploy/kustomization.yaml
  ```

  b. Compile o pacote usando o seguinte comando onde *<bundle.yaml>* está `bundle_pre_1_25.yaml` ou `bundle_post_1_25.yaml` baseado na sua versão do Kubernetes.

  ```
  kubectl kustomize deploy/ > deploy/<bundle.yaml>
  ```

Passos

1. Crie os recursos e implante o operador:

```
kubectl create -f deploy/<bundle.yaml>
```

2. Verifique se o operador, a implantação e as replicasets foram criados.

```
kubectl get all -n <operator-namespace>
```

Deve haver apenas **uma instância** do operador em um cluster do Kubernetes. Não crie várias implantações do operador Trident.

Passo 5: Atualize a localização do registo de imagens no `TridentOrchestrator`

O "Imagens de Trident e CSI" pode estar localizado em um Registro ou Registros diferentes, mas todas as imagens CSI devem estar localizadas no mesmo Registro. Atualize `deploy/crds/tridentorchestrator_cr.yaml` para adicionar as especificações de localização adicionais com base na configuração do seu registo.

Imagens em um Registro

```
imageRegistry: "<your-registry>"
autosupportImage: "<your-registry>/trident-autosupport:26.10"
tridentImage: "<your-registry>/trident:26.10.1"
```

Imagens em diferentes registos

```
imageRegistry: "<your-registry>"
autosupportImage: "<your-registry>/trident-autosupport:26.10"
tridentImage: "<your-registry>/trident:26.10.1"
```

Passo 6: Crie o `TridentOrchestrator` e instale o Trident

Agora você pode criar e instalar o `TridentOrchestrator` Trident. Opcionalmente, você pode usar ainda mais "Personalize a instalação do Trident"os atributos na `TridentOrchestrator` especificação. O exemplo a seguir mostra uma instalação onde as imagens Trident e CSI estão localizadas em diferentes Registros.

```
kubectl create -f deploy/crds/tridentorchestrator_cr.yaml
tridentorchestrator.trident.netapp.io/trident created

kubectl describe torc trident

Name:         trident
Namespace:
Labels:       <none>
Annotations: <none>
API Version: trident.netapp.io/v1
Kind:         TridentOrchestrator
...
Spec:
  Autosupport Image:  <your-registry>/trident-autosupport:26.10
  Debug:              true
  Image Registry:     <your-registry>
  Namespace:          trident
  Trident Image:      <your-registry>/trident:26.10.1
Status:
  Current Installation Params:
    IPv6:                       false
    Autosupport Hostname:
    Autosupport Image:          <your-registry>/trident-autosupport:26.10
    Autosupport Proxy:
    Autosupport Serial Number:
    Debug:                      true
    Http Request Timeout:       90s
    Image Pull Secrets:
    Image Registry:       <your-registry>
    k8sTimeout:           30
    Kubelet Dir:          /var/lib/kubelet
    Log Format:           text
    Probe Port:           17546
```

```
    Silence Autosupport:  false
    Trident Image:        <your-registry>/trident:26.10.1
  Message:                Trident installed
  Namespace:              trident
  Status:                 Installed
  Version:                v26.10.0
Events:
    Type Reason Age From Message ---- ------ ---- ---- -------Normal
    Installing 74s trident-operator.netapp.io Installing Trident Normal
    Installed 67s trident-operator.netapp.io Trident installed
```

Verifique a instalação

Existem várias maneiras de verificar sua instalação.

`TridentOrchestrator`Usando o status

O status de `TridentOrchestrator` indica se a instalação foi bem-sucedida e exibe a versão do Trident instalado. Durante a instalação, o status das `TridentOrchestrator` alterações de `Installing` para `Installed`. Se você observar o `Failed` status e o operador não conseguir recuperar sozinho, "verifique os logs".

Estado	Descrição
A instalar	O operador está a instalar o Trident utilizando este `TridentOrchestrator` CR.
Instalado	O Trident foi instalado com sucesso.
Desinstalação	O operador está desinstalando o Trident, porque `spec.uninstall=true`.
Desinstalado	O Trident é desinstalado.
Falha	O operador não pôde instalar, corrigir, atualizar ou desinstalar o Trident; o operador tentará recuperar automaticamente deste estado. Se este estado persistir, será necessário resolver o problema.
A atualizar	O operador está atualizando uma instalação existente.
Erro	O `TridentOrchestrator` não é utilizado. Outro já existe.

Usando o status de criação do pod

Você pode confirmar se a instalação do Trident foi concluída revisando o status dos pods criados:

```
kubectl get pods -n trident

NAME                                  READY   STATUS    RESTARTS
AGE
```

```
trident-controller-7d466bf5c7-v4cpw        6/6     Running   0
1m
trident-node-linux-mr6zc                   2/2     Running   0
1m
trident-node-linux-xrp7w                   2/2     Running   0
1m
trident-node-linux-zh2jt                   2/2     Running   0
1m
trident-operator-766f7b8658-ldzsv          1/1     Running   0
3m
```

Utilização `tridentctl`

Você pode usar `tridentctl` para verificar a versão do Trident instalada.

```
./tridentctl -n trident version

+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 26.10.1        | 26.10.1        |
+----------------+----------------+
```

Implantar operador Trident usando Helm (modo padrão)

Você pode implantar o operador Trident e instalar o Trident usando o Helm. Este processo aplica-se a instalações onde as imagens de contentor exigidas pelo Trident não são armazenadas em um Registro privado. Se tiver um registo de imagens privado, utilize o "processo para implantação off-line".

Informações críticas sobre Trident 26.10

Você deve ler as seguintes informações críticas sobre o Trident.

▼ informações críticas sobre o Trident

- O Trident 26.10 oferece suporte ao Kubernetes 1.27 a 1.37. Consulte a lista de imagens específicas da versão em "Requisitos da Trident" antes de atualizar o Kubernetes.
- O Trident reforça estritamente o uso de configuração multipathing em ambientes SAN, com um valor recomendado de `find_multipaths: no` no arquivo multipath.conf.

  O uso de configuração não multipathing ou o uso `find_multipaths: yes` de ou `find_multipaths: smart` valor no arquivo multipath.conf resultará em falhas de montagem. A Trident recomenda o uso de `find_multipaths: no` desde a versão 21,07.

Implante o operador Trident e instale o Trident usando o Helm

Usando o Trident"Carta do leme", você pode implantar o operador Trident e instalar o Trident em uma etapa.

Revise "a visão geral da instalação" para garantir que você atendeu aos pré-requisitos de instalação e selecionou a opção de instalação correta para o seu ambiente.

Antes de começar

Além do "pré-requisitos de implantação" você precisa de "Helm versão 3".

Passos

1. Adicione o repositório Helm do Trident:

```
helm repo add netapp-trident https://netapp.github.io/trident-helm-chart
```

2. Utilize `helm install` e especifique um nome para sua implantação, como no exemplo a seguir, onde `100.2610.0` é a versão do Trident que você está instalando.

```
helm install <name> netapp-trident/trident-operator --version 100.2610.0
--create-namespace --namespace <trident-namespace>
```

> Se você já criou um namespace para Trident, o `--create-namespace` parâmetro não criará um namespace adicional.

Você pode usar `helm list` para revisar detalhes de instalação, como nome, namespace, gráfico, status, versão do aplicativo e número de revisão.

Passe os dados de configuração durante a instalação

Há duas maneiras de passar dados de configuração durante a instalação:

Opção	Descrição
`--values` (ou `-f`)	Especifique um arquivo YAML com substituições. Isso pode ser especificado várias vezes e o arquivo mais à direita terá precedência.
`--set`	Especifique substituições na linha de comando.

Por exemplo, para alterar o valor padrão de `debug`, execute o seguinte comando onde `100.2610.0` é a versão do Trident que você está instalando:

```
helm install <name> netapp-trident/trident-operator --version 100.2610.0
--create-namespace --namespace trident --set tridentDebug=true
```

Opções de configuração

Esta tabela e o `values.yaml` arquivo, que faz parte do gráfico Helm, fornecem a lista de chaves e seus

valores padrão.

Opção	Descrição	Padrão
`nodeSelector`	Etiquetas de nó para atribuição de pod	
`podAnnotations`	Anotações do pod	
`deploymentAnnotations`	Anotações de implantação	
`tolerations`	Tolerâncias para atribuição de pod	

`affinity`	Afinidade para atribuição de pod	`affinity:`
		`  nodeAffinity:`
		
		`requiredDuringSchedulingIgnoredDuringExecution:`
		`  nodeSelectorTerms:`
		`  - matchExpressions:`
		`  - key: kubernetes.io/arch`
		`  operator: In`
		`  values:`
		`  - arm64`
		`  - amd64`
		`  - key: kubernetes.io/os`
		`  operator: In`
		`  values:`
		`  - linux`
		
		Não remova a afinidade padrão do arquivo values.yaml. Quando você quiser fornecer uma afinidade personalizada, estenda a afinidade padrão.

`tridentControllerPluginNodeSelector`	Seletores de nós adicionais para pods. Compreensão dos pods dos nós e dos pods do controladorConsulte para obter detalhes.	
`tridentControllerPluginTolerations`	Substitui as tolerâncias do Kubernetes para pods. Compreensão dos pods dos nós e dos pods do controladorConsulte para obter detalhes.	

Opção	Descrição	Padrão
`tridentNodePluginNodeSelector`	Seletores de nós adicionais para pods. Compreensão dos pods dos nós e dos pods do controladorConsulte para obter detalhes.	
`tridentNodePluginTolerations`	Substitui as tolerâncias do Kubernetes para pods. Compreensão dos pods dos nós e dos pods do controladorConsulte para obter detalhes.	
`imageRegistry`	Identifica o registo para as `trident-operator`, `trident`` e outras imagens. Deixe vazio para aceitar o padrão. IMPORTANTE: Ao instalar o Trident em um repositório privado, se você estiver usando `imageRegistry` o switch para especificar o local do repositório, não use `/netapp/` no caminho do repositório.	`""`
`imagePullPolicy`	Define a política de recebimento de imagens para o `trident-operator`.	`IfNotPresent`
`imagePullSecrets`	Define os segredos de extração da imagem para as `trident-operator`, `trident` e outras imagens.	
`kubeletDir`	Permite substituir a localização do host do estado interno do kubelet.	`"/var/lib/kubelet"`
`operatorLogLevel`	Permite que o nível de log do operador Trident seja definido como: `trace`, , `debug`, `info` `warn` , , `error` Ou `fatal`.	`"info"`
`operatorDebug`	Permite que o nível de log do operador Trident seja definido como debug.	`true`
`operatorImage`	Permite a substituição completa da imagem para `trident-operator`.	`""`
`operatorImageTag`	Permite substituir a etiqueta da `trident-operator` imagem.	`""`
`tridentIPv6`	Permite ativar o Trident para funcionar em clusters IPv6.	`false`

Opção	Descrição	Padrão
`tridentK8sTimeout`	Substitui o tempo limite padrão de 30 segundos para a maioria das operações da API do Kubernetes (se não for zero, em segundos).	`0`
`tridentHttpRequestTimeout`	Substitui o tempo limite padrão de 90 segundos para as solicitações HTTP, `0s` sendo uma duração infinita para o tempo limite. Valores negativos não são permitidos.	`"90s"`
`tridentSilenceAutosupport`	Permite desativar relatórios periódicos AutoSupport do Trident.	`false`
`tridentExcludeAutosupport`	Quando configurado para `true`, o Trident não configura o contêiner AutoSupport durante a instalação.	`false`
`tridentAutosupportImageTag`	Permite substituir a etiqueta da imagem para o contentor Trident AutoSupport.	`<version>`
`tridentAutosupportProxy`	Ativa o contentor Trident AutoSupport para telefonar para casa através de um proxy HTTP.	`""`
`tridentLogFormat`	Define o formato de registo Trident (`text`ou `json`).	`"text"`
`tridentDisableAuditLog`	Desativa o registador de auditoria Trident.	`true`
`tridentLogLevel`	Permite que o nível de log do Trident seja definido como: `trace`, , `debug`, `info` `warn` , , `error` `fatal` Ou .	`"info"`
`tridentDebug`	Permite definir o nível de registro do Trident para `debug` . Você pode automatizar o processo de desanexação forçada por meio da integração com o operador de verificação de integridade do nó (NHC). Para obter informações, consulte"Automatizando o failover de aplicações com estado usando o Trident." .	`false`
`tridentLogWorkflows`	Permite que fluxos de trabalho Trident específicos sejam ativados para registo de rastreio ou supressão de registos.	`""`

Opção	Descrição	Padrão
`tridentLogLayers`	Permite que camadas Trident específicas sejam ativadas para registo de rastreio ou supressão de registos.	`""`
`tridentImage`	Permite a substituição completa da imagem para o Trident.	`""`
`tridentImageTag`	Permite substituir a tag da imagem para Trident.	`""`
`tridentProbePort`	Permite substituir a porta padrão usada para sondas de disponibilidade/prontidão do Kubernetes.	`""`
`windows`	Permite que o Trident seja instalado no nó de trabalho do Windows.	`false`
`enableForceDetach`	Permite ativar a função forçar desanexar.	`false`
`enableDataLIFRefresh`	Atualiza os LIFs de dados NVMe para volumes publicados quando LIFs são adicionados, removidos ou recebem um novo endereço. Aplica-se a `ontap-san` com `sanType: nvme` no ONTAP unificado e a NVMe/TCP no ASA r2. Para obter mais informações, consulte "Atualizar LIFs de dados NVMe".	`false`
`excludePodSecurityPolicy`	Exclui a criação da diretiva de segurança do pod do operador.	`false`
`cloudProvider`	Defina como `"Azure"` ao usar identidades gerenciadas ou uma identidade na nuvem em um cluster AKS, `"AWS"` ao usar uma identidade na nuvem em um cluster EKS ou `"GCP"` ao usar identidade na nuvem no GKE ou a Federação de Identidade de Carga de Trabalho no OpenShift.	`""`

Opção	Descrição	Padrão
`cloudIdentity`	Defina como identidade da carga de trabalho ("azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxx") ao usar a identidade na nuvem em um cluster do AKS. Defina como função do AWS IAM ("'eks.amazonaws.com/role-arn: arn:aws:iam::123456:role/trident-role'") ao usar a identidade na nuvem em um cluster do EKS. Defina como a anotação de identidade na nuvem do GKE ao usar a identidade na nuvem no GKE. Não defina para a Federação de Identidade de Carga de Trabalho do OpenShift.	`""`
`gcpWIFAudience`	Público-alvo para o token de conta de serviço projetado usado com o Google Cloud Workload Identity Federation em OpenShift. Use o nome de recurso canônico do provedor. Para obter mais informações, consulte "Configure o Google Cloud NetApp Volumes com a federação de identidade da carga de trabalho".	`""`
`iscsiSelfHealingInterval`	O intervalo no qual a auto-recuperação iSCSI é invocada.	`5m0s`
`iscsiSelfHealingWaitTime`	A duração após a qual a auto-recuperação iSCSI inicia uma tentativa de resolver uma sessão obsoleta executando um logout e login subsequente.	`7m0s`
`nodePrep`	Permite que o Trident prepare os nós do cluster Kubernetes para gerenciar volumes usando o protocolo de armazenamento de dados especificado. **Atualmente, `iscsi` é o único valor suportado.** OBSERVAÇÃO: A partir do OpenShift 4.19, a versão mínima do Trident suportada para esse recurso é 25.06.1.	

Opção	Descrição	Padrão
`enableConcurrency`	Permite operações simultâneas do controlador do Trident para melhorar o desempenho. Para obter mais informações sobre os drivers compatíveis, consulte o xref:./trident-install/ Controller Scalability documentation.	falso
`k8sAPIQPS`	O limite de consultas por segundo (QPS) usado pelo controlador durante a comunicação com o servidor da API do Kubernetes. O valor de Burst é definido automaticamente com base no valor QPS.	`100`; opcional

Opção	Descrição	Padrão

`resources`	Define os limites de recursos e as solicitações do Kubernetes para o controlador Trident , o nó e os pods do operador. Você pode configurar CPU e memória para cada contêiner e sidecar para gerenciar a alocação de recursos no Kubernetes.
	
	Para obter mais informações sobre como configurar solicitações e limites de recursos, consulte"Gerenciamento de recursos para pods e contêineres" .
	
	• NÃO altere os nomes de nenhum contêiner ou campo.
	• NÃO altere o recuo - o recuo do YAML é fundamental para a análise correta.
	
	• Por padrão, não há limites aplicados; apenas as solicitações possuem valores padrão e são aplicadas automaticamente caso não sejam especificadas.
	• Os nomes dos contêineres são listados conforme aparecem nas especificações do pod.
	• Os sidecars estão listados abaixo de cada contêiner principal.
		<pre>resources:
		  controller:
		    trident-main:
		      requests:
		        cpu: 10m
		        memory: 80Mi
		      limits:
		          cpu:
		          memory:
		    csi-provisioner:
		      requests:
		        cpu: 2m
		        memory: 20Mi
		      limits:
		        cpu:
		        memory:
		    csi-attacher:
		      requests:
		        cpu: 2m
		        memory: 20Mi
		      limits:
		        cpu:
		        memory:
		    csi-resizer:
		      requests:
		        cpu: 3m
		        memory: 20Mi
		      limits:
		        cpu:
		        memory:
		    csi-snapshotter:
		      requests:
		        cpu: 2m
		        memory: 20Mi
		      limits:
		        cpu:
		        memory:
		    trident-autosupport:
		      requests:
		        cpu: 1m
		        memory: 30Mi
		      limits:
		        cpu:
		        memory:
		  node:</pre>

Opção	Descrição	Padrão
`httpsMetrics`	Habilite o HTTPS para o endpoint de métricas do Prometheus.	falso
`hostNetwork`	Habilita a conexão de rede do host para o controlador Trident . Isso é útil quando você deseja separar o tráfego de front-end do tráfego de back-end em uma rede com várias residências.	falso

Compreensão dos pods dos nós e dos pods do controlador

O Trident é executado como um único pod de controlador, além de um pod de nó em cada nó de trabalho no cluster. O pod de nó deve estar em execução em qualquer host onde você queira montar um volume Trident potencialmente.

Kubernetes "seletores de nós" e "tolerações e taints"são usados para restringir um pod a ser executado em um nó específico ou preferencial. Usando o ControllerPlugin e `NodePlugin`o , você pode especificar restrições e substituições.

- O plugin controlador lida com o provisionamento e gerenciamento de volume, como snapshots e redimensionamento.
- O plug-in do nó manipula a conexão do armazenamento ao nó.

Implantar operador Trident usando Helm (modo off-line)

Você pode implantar o operador Trident e instalar o Trident usando o Helm. Este processo aplica-se a instalações onde as imagens de contentor exigidas pelo Trident são armazenadas em um Registro privado. Se não tiver um registo de imagens privado, utilize o "processo para implantação padrão".

Informações críticas sobre Trident 26.10

Você deve ler as seguintes informações críticas sobre o Trident.

**▼ informações críticas sobre o Trident **

- O Trident 26.10 oferece suporte ao Kubernetes 1.27 a 1.37. Consulte a lista de imagens específicas da versão em "Requisitos da Trident" antes de atualizar o Kubernetes.
- O Trident reforça estritamente o uso de configuração multipathing em ambientes SAN, com um valor recomendado de `find_multipaths: no` no arquivo multipath.conf.

  O uso de configuração não multipathing ou o uso `find_multipaths: yes` de ou `find_multipaths: smart` valor no arquivo multipath.conf resultará em falhas de montagem. A Trident recomenda o uso de `find_multipaths: no` desde a versão 21,07.

Implante o operador Trident e instale o Trident usando o Helm

Usando o Trident"Carta do leme", você pode implantar o operador Trident e instalar o Trident em uma etapa.

Revise "a visão geral da instalação" para garantir que você atendeu aos pré-requisitos de instalação e selecionou a opção de instalação correta para o seu ambiente.

Antes de começar

Além do "pré-requisitos de implantação" você precisa de "Helm versão 3".

> Ao instalar o Trident em um repositório privado, se você estiver usando o `imageRegistry` switch para especificar o local do repositório, não use `/netapp/` no caminho do repositório.

Passos

1. Adicione o repositório Helm do Trident:

```
helm repo add netapp-trident https://netapp.github.io/trident-helm-chart
```

2. Use `helm install` e especifique um nome para sua implantação e o local do registro de imagens. Seu "Imagens de Trident e CSI" pode estar localizado em um registro ou em registros diferentes, mas todas as imagens CSI devem estar no mesmo registro. Nos exemplos, `100.2610.0` é a versão do Trident que você está instalando.

Imagens em um Registro

```
helm install <name> netapp-trident/trident-operator --version
100.2610.0 --set imageRegistry=<your-registry> --create-namespace
--namespace <trident-namespace> --set nodePrep={iscsi}
```

Imagens em diferentes registos

```
helm install <name> netapp-trident/trident-operator --version
100.2610.0 --set imageRegistry=<your-registry> --set operatorImage
=<your-registry>/trident-operator:26.10.1 --set
tridentAutosupportImage=<your-registry>/trident-autosupport:26.10
--set tridentImage=<your-registry>/trident:26.10.1 --create
-namespace --namespace <trident-namespace> --set nodePrep={iscsi}
```

> Se você já criou um namespace para Trident, o `--create-namespace` parâmetro não criará um namespace adicional.

Você pode usar `helm list` para revisar detalhes de instalação, como nome, namespace, gráfico, status, versão do aplicativo e número de revisão.

Passe os dados de configuração durante a instalação

Há duas maneiras de passar dados de configuração durante a instalação:

Opção	Descrição
`--values` (ou `-f`)	Especifique um arquivo YAML com substituições. Isso pode ser especificado várias vezes e o arquivo mais à direita terá precedência.
`--set`	Especifique substituições na linha de comando.

Por exemplo, para alterar o valor padrão de `debug`, execute o seguinte comando onde `100.2610.0` é a versão do Trident que você está instalando:

```
helm install <name> netapp-trident/trident-operator --version 100.2610.0
--create-namespace --namespace trident --set tridentDebug=true
```

Para adicionar o valor nodePrep, execute o seguinte comando:

```
helm install <name> netapp-trident/trident-operator --version 100.2610.0
--create-namespace --namespace trident --set nodePrep={iscsi}
```

Opções de configuração

Esta tabela e o `values.yaml` arquivo, que faz parte do gráfico Helm, fornecem a lista de chaves e seus valores padrão.

> Não remova a afinidade padrão do arquivo values.yaml. Quando você quiser fornecer uma afinidade personalizada, estenda a afinidade padrão.

Opção	Descrição	Padrão
`nodeSelector`	Etiquetas de nó para atribuição de pod	
`podAnnotations`	Anotações do pod	
`deploymentAnnotations`	Anotações de implantação	
`tolerations`	Tolerâncias para atribuição de pod	

Opção	Descrição	Padrão

`affinity`	Afinidade para atribuição de pod	`affinity:`
		`  nodeAffinity:`
		
		`requiredDuringSchedul`
		`ingIgnoredDuringExecu`
		`tion:`
		
		`nodeSelectorTerms:`
		`  -`
		`matchExpressions:`
		`  - key:`
		`kubernetes.io/arch`
		
		`operator: In`
		`  values:`
		`  - arm64`
		`  - amd64`
		`  - key:`
		`kubernetes.io/os`
		
		`operator: In`
		`  values:`
		`  - linux`
		
		Não remova a afinidade padrão do arquivo values.yaml. Quando você quiser fornecer uma afinidade personalizada, estenda a afinidade padrão.

`tridentControllerPluginNod` `eSelector`	Seletores de nós adicionais para pods. "Compreensão dos pods dos nós e dos pods do controlador"Consulte para obter detalhes.	
`tridentControllerPluginTol` `erations`	Substitui as tolerâncias do Kubernetes para pods. "Compreensão dos pods dos nós e dos pods do controlador"Consulte para obter detalhes.	

Opção	Descrição	Padrão
`tridentNodePluginNodeSelector`	Seletores de nós adicionais para pods. "Compreensão dos pods dos nós e dos pods do controlador"Consulte para obter detalhes.	
`tridentNodePluginTolerations`	Substitui as tolerâncias do Kubernetes para pods. "Compreensão dos pods dos nós e dos pods do controlador"Consulte para obter detalhes.	
`imageRegistry`	Identifica o registo para as `trident-operator`, `trident`` e outras imagens. Deixe vazio para aceitar o padrão. IMPORTANTE: Ao instalar o Trident em um repositório privado, se você estiver usando `imageRegistry` o switch para especificar o local do repositório, não use `/netapp/` no caminho do repositório.	""
`imagePullPolicy`	Define a política de recebimento de imagens para o `trident-operator`.	`IfNotPresent`
`imagePullSecrets`	Define os segredos de extração da imagem para as `trident-operator`, `trident` e outras imagens.	
`kubeletDir`	Permite substituir a localização do host do estado interno do kubelet.	`"/var/lib/kubelet"`
`operatorLogLevel`	Permite que o nível de log do operador Trident seja definido como: `trace`, , `debug`, `info` `warn`, , `error` Ou `fatal`.	`"info"`
`operatorDebug`	Permite que o nível de log do operador Trident seja definido como debug.	`true`
`operatorImage`	Permite a substituição completa da imagem para `trident-operator`.	""
`operatorImageTag`	Permite substituir a etiqueta da `trident-operator` imagem.	""
`tridentIPv6`	Permite ativar o Trident para funcionar em clusters IPv6.	`false`

Opção	Descrição	Padrão
`tridentK8sTimeout`	Substitui o tempo limite padrão de 180 segundos para a maioria das operações da API do Kubernetes (se diferente de zero, em segundos). O `tridentK8sTimeout` O parâmetro é aplicável somente para instalação do Trident .	`180`
`tridentHttpRequestTimeout`	Substitui o tempo limite padrão de 90 segundos para as solicitações HTTP, `0s` sendo uma duração infinita para o tempo limite. Valores negativos não são permitidos.	`"90s"`
`tridentSilenceAutosupport`	Permite desativar relatórios periódicos AutoSupport do Trident.	`false`
`tridentExcludeAutosupport`	Quando configurado para `true`, o Trident não configura o contêiner AutoSupport durante a instalação.	`false`
`tridentAutosupportImageTag`	Permite substituir a etiqueta da imagem para o contentor Trident AutoSupport.	`<version>`
`tridentAutosupportProxy`	Ativa o contentor Trident AutoSupport para telefonar para casa através de um proxy HTTP.	""
`tridentLogFormat`	Define o formato de registo Trident (`text`ou `json`).	`"text"`
`tridentDisableAuditLog`	Desativa o registador de auditoria Trident.	`true`
`tridentLogLevel`	Permite que o nível de log do Trident seja definido como: `trace`, , `debug`, `info` `warn` , , `error` `fatal` Ou .	`"info"`
`tridentDebug`	Permite que o nível de log do Trident seja definido como `debug`.	`false`
`tridentLogWorkflows`	Permite que fluxos de trabalho Trident específicos sejam ativados para registo de rastreio ou supressão de registos.	""

Opção	Descrição	Padrão
`tridentLogLayers`	Permite que camadas Trident específicas sejam ativadas para registo de rastreio ou supressão de registos.	""
`tridentImage`	Permite a substituição completa da imagem para o Trident.	""
`tridentImageTag`	Permite substituir a tag da imagem para Trident.	""
`tridentProbePort`	Permite substituir a porta padrão usada para sondas de disponibilidade/prontidão do Kubernetes.	""
`windows`	Permite que o Trident seja instalado no nó de trabalho do Windows.	`false`
`enableForceDetach`	Permite ativar a função de desprendimento forçado. Você pode automatizar o processo de desanexação forçada por meio da integração com o operador de verificação de integridade do nó (NHC). Para obter informações, consulte"Automatizando o failover de aplicações com estado usando o Trident." .	`false`
`enableDataLIFRefresh`	Atualiza os LIFs de dados NVMe para volumes publicados quando LIFs são adicionados, removidos ou recebem um novo endereço. Aplica-se a `ontap-san` com `sanType: nvme` no ONTAP unificado e a NVMe/TCP no ASA r2. Para obter mais informações, consulte "Atualizar LIFs de dados NVMe".	`false`
`excludePodSecurityPolicy`	Exclui a criação da diretiva de segurança do pod do operador.	`false`

Opção	Descrição	Padrão
`nodePrep`	Permite que o Trident prepare os nós do cluster do Kubernetes para gerenciar volumes usando o protocolo de storage de dados especificado. **Atualmente, `iscsi` é o único valor suportado.** A partir do OpenShift 4.19, a versão mínima do Trident suportada para esse recurso é 25.06.1.	

Opção	Descrição	Padrão

`resources`	Define os limites de recursos e as solicitações do Kubernetes para o controlador Trident , o nó e os pods do operador. Você pode configurar CPU e memória para cada contêiner e sidecar para gerenciar a alocação de recursos no Kubernetes.
	
	Para obter mais informações sobre como configurar solicitações e limites de recursos, consulte"Gerenciamento de recursos para pods e contêineres" .
	
	• NÃO altere os nomes de nenhum contêiner ou campo.
	• NÃO altere o recuo - o recuo do YAML é fundamental para a análise correta.
	
	• Por padrão, não há limites aplicados; apenas as solicitações possuem valores padrão.
	• Os nomes dos contêineres são listados conforme aparecem nas especificações do pod.
	• Os sidecars estão listados abaixo de cada contêiner principal.
	• Verifique o TORC. `status.CurrentInstallationParams`
		`resources:`
		`  controller:`
		`  trident-main:`
		`  requests:`
		`  cpu: 10m`
		`  memory: 80Mi`
		`  limits:`
		`  cpu:`
		`  memory:`
		`  csi-provisioner:`
		`  requests:`
		`  cpu: 2m`
		`  memory: 20Mi`
		`  limits:`
		`  cpu:`
		`  memory:`
		`  csi-attacher:`
		`  requests:`
		`  cpu: 2m`
		`  memory: 20Mi`
		`  limits:`
		`  cpu:`
		`  memory:`
		`  csi-resizer:`
		`  requests:`
		`  cpu: 3m`
		`  memory: 20Mi`
		`  limits:`
		`  cpu:`
		`  memory:`
		`  csi-snapshotter:`
		`  requests:`
		`  cpu: 2m`
		`  memory: 20Mi`
		`  limits:`
		`  cpu:`
		`  memory:`
		`  trident-autosupport:`
		`  requests:`
		`  cpu: 1m`
		`  memory: 30Mi`
		`  limits:`
		`  cpu:`
		`  memory:`

Personalizar a instalação do operador Trident

O operador Trident permite personalizar a instalação do Trident usando os atributos na `TridentOrchestrator` especificação. Se você quiser personalizar a instalação além do que `TridentOrchestrator` os argumentos permitem, considere usar `tridentctl` para gerar manifestos YAML personalizados para modificar conforme necessário.

Compreensão dos pods dos nós e dos pods do controlador

O Trident é executado como um único pod de controlador e um pod de nó em cada nó de trabalho do cluster. O pod do nó deve estar em execução em qualquer host onde você queira potencialmente montar um volume Trident .

Kubernetes "seletores de nós" e "tolerações e taints"são usados para restringir um pod a ser executado em um nó específico ou preferencial. Usando o ControllerPlugin e `NodePlugin`o , você pode especificar restrições e substituições.

- O plugin controlador lida com o provisionamento e gerenciamento de volume, como snapshots e redimensionamento.
- O plug-in do nó manipula a conexão do armazenamento ao nó.

Opções de configuração

> `spec.namespace` É especificado em `TridentOrchestrator` para indicar o namespace onde o Trident está instalado. Este parâmetro **não pode ser atualizado após a instalação do Trident**. Tentar fazê-lo faz com que o `TridentOrchestrator` status mude para `Failed`. O Trident não se destina a ser migrado entre namespaces.

Esta tabela detalha `TridentOrchestrator` atributos.

Parâmetro	Descrição	Padrão
`namespace`	Namespace para instalar o Trident em	`"default"`
`debug`	Ativar depuração para Trident	`false`
`enableForceDetach`	`ontap-san`, `ontap-san-economy` , `ontap-nas` , e `ontap-nas-economy` apenas. Funciona com o recurso Non-Graceful Node Shutdown (NGNS) do Kubernetes para permitir que os administradores do cluster migrem cargas de trabalho com volumes montados para novos nós com segurança, caso um nó apresente problemas. Para obter informações, consulte"Automatizando o failover de aplicações com estado usando o Trident." .	`false`
`enableDataLIFRefresh`	Permite a atualização das LIFs de dados NVMe para volumes publicados quando LIFs são adicionadas, removidas ou recebem um novo endereço. Aplica-se a `ontap-san` com `sanType: nvme` no ONTAP unificado e ao NVMe/TCP no ASA r2. Para mais informações, consulte "Atualizar LIFs de dados NVMe".	`false`

Parâmetro	Descrição	Padrão
`windows`	A configuração para `true` permite a instalação em nós de trabalho do Windows.	`false`
`cloudProvider`	Defina como `"Azure"` quando utilizar identidades geridas ou uma identidade de nuvem num cluster AKS. Defina como `"AWS"` ao usar uma identidade de nuvem em um cluster EKS. Defina como `"GCP"` quando estiver usando uma identidade de nuvem em um cluster GKE.	`""`
`cloudIdentity`	Defina como identidade da carga de trabalho ("azure.Workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxxxxxxx") ao usar identidade da nuvem em um cluster AKS. Definido como função do AWS IAM ("'eks.amazonaws.com/role-arn: arn:aws:iam::123456:role/Trident-role'") ao usar a identidade da nuvem em um cluster do EKS. Defina como identidade de nuvem ("'iam.gke.io/gcp-service-account: xxxx@mygcpproject.iam.gserviceaccount.com'") ao usar identidade de nuvem em um cluster GKE.	`""`
`gcpWIFAudience`	Público-alvo para o token de conta de serviço projetado usado com o Google Cloud Workload Identity Federation em OpenShift. Use o nome de recurso canônico do provedor. Para obter mais informações, consulte "Configure o Google Cloud NetApp Volumes com a federação de identidade da carga de trabalho".	`""`
`IPv6`	Instale o Trident em IPv6	falso
`k8sTimeout`	Tempo limite para operações do Kubernetes. O `k8sTimeout` O parâmetro é aplicável somente para instalação do Trident .	`180sec`
`silenceAutosupport`	Não envie pacotes AutoSupport para o NetApp automaticamente	`false`
`autosupportImage`	A imagem do recipiente para a telemetria AutoSupport	`"netapp/trident-autosupport:26.10"`
`autosupportProxy`	O endereço/porta de um proxy para o envio de telemetria AutoSupport	`"http://proxy.example.com:8888"`
`uninstall`	Um sinalizador usado para desinstalar o Trident	`false`
`logFormat`	Formato de Registro Trident a ser usado [text,json]	`"text"`
`tridentImage`	Imagem Trident a instalar	`"netapp/trident:26.10.1"`
`imageRegistry`	Caminho para o Registro interno, do formato `<registry FQDN>[:port][/subpath]`	`"registry.k8s.io"`

Parâmetro	Descrição	Padrão
`kubeletDir`	Caminho para o diretório kubelet no host	`"/var/lib/kubelet"`
`wipeout`	Uma lista de recursos a serem excluídos para realizar uma remoção completa do Trident	
`imagePullSecrets`	Segredos para extrair imagens de um Registro interno	
`imagePullPolicy`	Define a política de recebimento de imagens para o operador Trident. Os valores válidos são: `Always` Para sempre puxar a imagem. `IfNotPresent` para puxar a imagem apenas se ela ainda não existir no nó. `Never` para nunca puxar a imagem.	`IfNotPresent`
`controllerPluginNodeSelector`	Seletores de nós adicionais para pods. Segue o mesmo formato que `pod.spec.nodeSelector`.	Sem padrão; opcional
`controllerPluginTolerations`	Substitui as tolerâncias do Kubernetes para pods. Segue o mesmo formato que `pod.spec.Tolerations`.	Sem padrão; opcional
`nodePluginNodeSelector`	Seletores de nós adicionais para pods. Segue o mesmo formato que `pod.spec.nodeSelector`.	Sem padrão; opcional
`nodePluginTolerations`	Substitui as tolerâncias do Kubernetes para pods. Segue o mesmo formato que `pod.spec.Tolerations`.	Sem padrão; opcional
`nodePrep`	Permite que o Trident prepare os nós do cluster do Kubernetes para gerenciar volumes usando o protocolo de storage de dados especificado. **Atualmente, `iscsi` é o único valor suportado.** A partir do OpenShift 4.19, a versão mínima do Trident suportada para esse recurso é 25.06.1.	
`k8sAPIQPS`	O limite de consultas por segundo (QPS) usado pelo controlador durante a comunicação com o servidor da API do Kubernetes. O valor de Burst é definido automaticamente com base no valor QPS.	`100`; opcional
`enableConcurrency`	Permite operações simultâneas do controlador do Trident para melhorar o desempenho. Para obter mais informações sobre os drivers compatíveis, consulte o xref:./trident-install/ Controller Scalability documentation.	`enableCOSI`

Parâmetro	Descrição	Padrão
Habilita a interface de storage de objetos de contêiner (COSI) do Trident. Exige `enableConcurrency` que seja definido como `true`. O Trident COSI não pode ser executado em conjunto com o controlador COSI da comunidade. Para obter mais informações, consulte "Provisionar storage de objetos com COSI".	`false`	`resources`

Parâmetro	Descrição	Padrão

Define os limites de recursos e as solicitações do Kubernetes para o controlador Trident e os pods dos nós. Você pode configurar CPU e memória para cada contêiner e sidecar para gerenciar a alocação de recursos no Kubernetes.
 	
Para obter mais informações sobre como configurar solicitações e limites de recursos, consulte"Gerenciamento de recursos para pods e contêineres" .
 	
• NÃO altere os nomes de nenhum contêiner ou campo.
• NÃO altere o recuo - o recuo do YAML é fundamental para a análise correta.
 	
• Por padrão, não há limites aplicados; apenas as solicitações possuem
	<pre>resources:
	  controller:
	    trident-main:
	      requests:
	        cpu: 10m
	        memory: 80Mi
	      limits:
	          cpu:
	          memory:
	    csi-provisioner:
	      requests:
	        cpu: 2m
	        memory: 20Mi
	      limits:
	        cpu:
	        memory:
	    csi-attacher:
	      requests:
	        cpu: 2m
	        memory: 20Mi
	      limits:
	        cpu:
	        memory:
	    csi-resizer:
	      requests:
	        cpu: 3m
	        memory: 20Mi
	      limits:
	        cpu:
	        memory:
	    csi-snapshotter:
	      requests:
	        cpu: 2m
	        memory: 20Mi
	      limits:
	        cpu:
	        memory:
	    trident-autosupport:
	      requests:
	        cpu: 1m
	        memory: 30Mi
	      limits:
	        cpu:
	        memory:
	  node:</pre>
		`httpsMetrics`

Parâmetro	Descrição	Padrão
Habilite o HTTPS para o endpoint de métricas do Prometheus.	falso	`hostNetwork`

Para obter mais informações sobre a formatação dos parâmetros do pod, "Atribuindo pods a nós"consulte .

Exemplos de configurações

Você pode usar os atributos emOpções de configuração ao definir `TridentOrchestrator` Para personalizar sua instalação.

▼ **Configuração personalizada básica**

Este exemplo foi criado após a execução do `cat deploy/crds/tridentorchestrator_cr_imagepullsecrets.yaml` O comando representa uma instalação personalizada básica:

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  imagePullSecrets:
  - thisisasecret
```

▼ **Seletores de nós**

Este exemplo instala o Trident com seletores de nós.

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  controllerPluginNodeSelector:
    nodetype: master
  nodePluginNodeSelector:
    storage: netapp
```

▼ **nós de trabalho do Windows**

Este exemplo foi criado após a execução do `cat deploy/crds/tridentorchestrator_cr.yaml` O

comando instala o Trident em um nó de trabalho do Windows.

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  windows: true
```

▼ **Identidades gerenciadas em um cluster AKS**

Este exemplo instala o Trident para habilitar identidades gerenciadas em um cluster AKS.

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  cloudProvider: "Azure"
```

▼ **Identidade na nuvem em um cluster AKS**

Este exemplo instala o Trident para uso com uma identidade na nuvem em um cluster AKS.

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  cloudProvider: "Azure"
  cloudIdentity: 'azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-
xxxx-xxxxxxxxxxx'
```

▼ **Identidade na nuvem em um cluster EKS**

Este exemplo instala o Trident para uso com uma identidade na nuvem em um cluster AKS.

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
```

```
  name: trident
spec:
  debug: true
  namespace: trident
  cloudProvider: "AWS"
  cloudIdentity: "'eks.amazonaws.com/role-arn:
arn:aws:iam::123456:role/trident-role'"
```

▼ **Identidade de nuvem para GKE**

Este exemplo instala o Trident para uso com uma identidade na nuvem em um cluster GKE.

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-gcp-gcnv
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: '012345678901'
  network: gcnv-network
  location: us-west2
  serviceLevel: Premium
  storagePool: pool-premium1
```

▼ **Configuração de solicitações e limites de recursos do Kubernetes para o controlador Trident e pods de nós Linux do Trident .**

Este exemplo configura as solicitações e limites de recursos do Kubernetes para os pods do controlador Trident e do nó Linux do Trident .



Aviso: Os valores de solicitação e limite fornecidos neste exemplo são apenas para fins de demonstração. Ajuste esses valores com base no seu ambiente e nas necessidades da sua carga de trabalho.

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  imagePullSecrets:
  - thisisasecret
  resources:
    controller:
      trident-main:
```

```
        requests:
          cpu: 10m
          memory: 80Mi
        limits:
          cpu: 200m
          memory: 256Mi
      # sidecars
      csi-provisioner:
        requests:
          cpu: 2m
          memory: 20Mi
        limits:
          cpu: 100m
          memory: 64Mi
      csi-attacher:
        requests:
          cpu: 2m
          memory: 20Mi
        limits:
          cpu: 100m
          memory: 64Mi
      csi-resizer:
        requests:
          cpu: 3m
          memory: 20Mi
        limits:
          cpu: 100m
          memory: 64Mi
      csi-snapshotter:
        requests:
          cpu: 2m
          memory: 20Mi
        limits:
          cpu: 100m
          memory: 64Mi
      trident-autosupport:
        requests:
          cpu: 1m
          memory: 30Mi
        limits:
          cpu: 50m
          memory: 128Mi
    node:
      linux:
        trident-main:
          requests:
```

```
            cpu: 10m
            memory: 60Mi
          limits:
            cpu: 200m
            memory: 256Mi
        # sidecars
        node-driver-registrar:
          requests:
            cpu: 1m
            memory: 10Mi
          limits:
            cpu: 50m
            memory: 32Mi
```

▼ Configuração de solicitações e limites de recursos do Kubernetes para o controlador Trident e pods de nós Trident Windows e Linux.

Este exemplo configura solicitações e limites de recursos do Kubernetes para o controlador Trident e para os pods dos nós Trident Windows e Linux.



Aviso: Os valores de solicitação e limite fornecidos neste exemplo são apenas para fins de demonstração. Ajuste esses valores com base no seu ambiente e nas necessidades da sua carga de trabalho.

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  imagePullSecrets:
  - thisisasecret
  windows: true
  resources:
    controller:
      trident-main:
        requests:
          cpu: 10m
          memory: 80Mi
        limits:
          cpu: 200m
          memory: 256Mi
        # sidecars
      csi-provisioner:
        requests:
          cpu: 2m
```

```
          memory: 20Mi
        limits:
          cpu: 100m
          memory: 64Mi
      csi-attacher:
        requests:
          cpu: 2m
          memory: 20Mi
        limits:
          cpu: 100m
          memory: 64Mi
      csi-resizer:
        requests:
          cpu: 3m
          memory: 20Mi
        limits:
          cpu: 100m
          memory: 64Mi
      csi-snapshotter:
        requests:
          cpu: 2m
          memory: 20Mi
        limits:
          cpu: 100m
          memory: 64Mi
      trident-autosupport:
        requests:
          cpu: 1m
          memory: 30Mi
        limits:
          cpu: 50m
          memory: 128Mi
    node:
      linux:
        trident-main:
          requests:
            cpu: 10m
            memory: 60Mi
          limits:
            cpu: 200m
            memory: 256Mi
        # sidecars
        node-driver-registrar:
          requests:
            cpu: 1m
            memory: 10Mi
```

```
          limits:
            cpu: 50m
            memory: 32Mi
      windows:
        trident-main:
          requests:
            cpu: 6m
            memory: 40Mi
          limits:
            cpu: 200m
            memory: 128Mi
        # sidecars
        node-driver-registrar:
          requests:
            cpu: 6m
            memory: 40Mi
          limits:
            cpu: 100m
            memory: 128Mi
        liveness-probe:
          requests:
            cpu: 2m
            memory: 40Mi
          limits:
            cpu: 50m
            memory: 64Mi
```

Instale usando o tridentctl

Instale o Trident usando o tridentctl

Você pode instalar o Trident usando `tridentctl`o . Este processo aplica-se a instalações onde as imagens de contentor exigidas pelo Trident são armazenadas em um Registro privado ou não. Para personalizar sua `tridentctl` implantação, "Personalizar a implantação do tridentctl"consulte .

Informações críticas sobre Trident 26.10

Você deve ler as seguintes informações críticas sobre o Trident.

▼ ** informações críticas sobre o Trident **

- O Kubernetes 1.37 é compatível com Trident 26.10. Atualize o Trident antes de atualizar o Kubernetes.
- O Trident reforça estritamente o uso de configuração multipathing em ambientes SAN, com um valor recomendado de `find_multipaths: no` no arquivo multipath.conf.

  O uso de configuração não multipathing ou o uso `find_multipaths: yes` de ou

`find_multipaths: smart` valor no arquivo multipath.conf resultará em falhas de montagem. A Trident recomenda o uso de `find_multipaths: no` desde a versão 21,07.

Instale o Trident usando `tridentctl`

Revise "a visão geral da instalação" para garantir que você atendeu aos pré-requisitos de instalação e selecionou a opção de instalação correta para o seu ambiente.

Antes de começar

Antes de iniciar a instalação, faça login no host Linux e verifique se ele está gerenciando um sistema operacional em funcionamento "Cluster compatível com Kubernetes" e se você possui os privilégios necessários.

> Com OpenShift, use `oc` em vez de `kubectl` em todos os exemplos que se seguem, e faça login como **system:admin** primeiro executando `oc login -u system:admin` ou `oc login -u kube-admin`.

▼ **Details**

1. Verifique sua versão do Kubernetes:

```
kubectl version
```

2. Verifique o Privileges do administrador do cluster:

```
kubectl auth can-i '*' '*' --all-namespaces
```

3. Verifique se você pode iniciar um pod que usa uma imagem do Docker Hub e alcançar seu sistema de armazenamento pela rede de pod:

```
kubectl run -i --tty ping --image=busybox --restart=Never --rm -- \
  ping <management IP>
```

Passo 1: Baixe o pacote de instalação do Trident

O pacote de instalação do Trident cria um pod Trident, configura os objetos CRD que são usados para manter seu estado e inicializa os sidecars CSI para executar ações como provisionar e anexar volumes aos hosts do cluster. Baixe e extraia a versão mais recente do instalador do Trident a partir "A seção *assets* no GitHub"do . Atualize _ cliente Trident-installer-XX.XX.X.tar.gz>_ no exemplo com a versão Trident selecionada.

```
wget https://github.com/NetApp/trident/releases/download/v26.10.0/trident-
installer-26.10.1.tar.gz
tar -xf trident-installer-26.10.1.tar.gz
cd trident-installer
```

Passo 2: Instale o Trident

Instale o Trident no namespace desejado executando o `tridentctl install` comando. Você pode adicionar argumentos adicionais para especificar a localização do Registro da imagem.

Modo padrão

```
./tridentctl install -n trident
```

Imagens em um Registro

```
./tridentctl install -n trident --image-registry <your-registry>
--autosupport-image <your-registry>/trident-autosupport:26.10 --trident
-image <your-registry>/trident:26.10.1
```

Imagens em diferentes registos

```
./tridentctl install -n trident --image-registry <csi-registry>
--autosupport-image <autosupport-registry>/trident-autosupport:26.10
--trident-image <trident-registry>/trident:26.10.1
```

Seu status de instalação deve ser parecido com isso.

```
....
INFO Starting Trident installation.                namespace=trident
INFO Created service account.
INFO Created cluster role.
INFO Created cluster role binding.
INFO Added finalizers to custom resource definitions.
INFO Created Trident service.
INFO Created Trident secret.
INFO Created Trident deployment.
INFO Created Trident daemonset.
INFO Waiting for Trident pod to start.
INFO Trident pod started.                          namespace=trident
pod=trident-controller-679648bd45-cv2mx
INFO Waiting for Trident REST interface.
INFO Trident REST interface is up.                 version=26.10.1
INFO Trident installation succeeded.
....
```

Verifique a instalação

Você pode verificar sua instalação usando o status de criação do pod ou `tridentctl`.

Usando o status de criação do pod

Você pode confirmar se a instalação do Trident foi concluída revisando o status dos pods criados:

```
kubectl get pods -n trident

NAME                                READY   STATUS    RESTARTS   AGE
trident-controller-679648bd45-cv2mx   6/6     Running   0          5m29s
trident-node-linux-vgc8n              2/2     Running   0          5m29s
```

Se o instalador não for concluído com êxito ou `trident-controller-<generated id>` não estiver com o status **Running**, a plataforma não foi instalada. Use `-d` para "ative o modo de depuração" e solucionar o problema.

Utilização `tridentctl`

Você pode usar `tridentctl` para verificar a versão do Trident instalada.

```
./tridentctl -n trident version

+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 26.10.1        | 26.10.1        |
+----------------+----------------+
```

Exemplos de configurações

Os exemplos a seguir fornecem exemplos de configurações para a instalação do Trident usando `tridentctl`o .

▼ **Nós do Windows**

Para permitir que o Trident seja executado em nós do Windows:

```
tridentctl install --windows -n trident
```

▼ **Força a soltar**

Para obter informações, consulte"Automatizando o failover de aplicações com estado usando o Trident." .

```
tridentctl install --enable-force-detach=true -n trident
```

▼ **Habilitar operações simultâneas do controlador Trident**

Para habilitar operações simultâneas do controlador do Trident e melhorar o desempenho, adicione a opção `--enable-concurrency` durante a instalação, conforme mostrado neste exemplo. Para obter mais informações sobre os drivers compatíveis, consulte o xref:./trident-install/ Controller Scalability

documentation.

```
tridentctl install --enable-concurrency -n trident
```

▼ **Atualizar LIFs de dados NVMe**

Para atualizar as LIFs de dados NVMe para volumes publicados quando a configuração da LIF for alterada, instale o Trident com `--enable-datalif-refresh`. Para obter detalhes sobre os drivers e a configuração compatíveis, consulte "Atualizar LIFs de dados NVMe".

```
tridentctl install --enable-datalif-refresh -n trident
```

Personalize a instalação do Trident tridentctl

Você pode usar o instalador do Trident para personalizar a instalação.

Saiba mais sobre o instalador

O instalador do Trident permite personalizar atributos. Por exemplo, se você tiver copiado a imagem Trident para um repositório privado, poderá especificar o nome da imagem `--trident-image` usando o . Se você copiou a imagem do Trident, bem como as imagens do sidecar do CSI necessárias para um repositório privado, pode ser preferível especificar a localização desse repositório usando o `--image-registry` switch, que assume o formulário `<registry FQDN>[:port]`.

> Ao instalar o Trident em um repositório privado, se você estiver usando o `--image-registry` switch para especificar o local do repositório, não use `/netapp/` no caminho do repositório. Por exemplo: `./tridentctl install --image-registry <image-registry> -n <namespace>`

Se você estiver usando uma distribuição do Kubernetes, onde `kubelet` mantém seus dados em um caminho diferente do habitual `/var/lib/kubelet`, você poderá especificar o caminho alternativo usando `--kubelet-dir`o .

Se você precisar personalizar a instalação além do que os argumentos do instalador permitem, você também pode personalizar os arquivos de implantação. Usando o `--generate-custom-yaml` parâmetro cria os seguintes arquivos YAML no diretório do instalador `setup`:

- `trident-clusterrolebinding.yaml`
- `trident-deployment.yaml`
- `trident-crds.yaml`
- `trident-clusterrole.yaml`
- `trident-daemonset.yaml`
- `trident-service.yaml`
- `trident-namespace.yaml`
- `trident-serviceaccount.yaml`

- `trident-resourcequota.yaml*`

Depois de gerar esses arquivos, você pode modificá-los de acordo com suas necessidades e, em seguida, usá-los `--use-custom-yaml` para instalar sua implantação personalizada.

```
./tridentctl install -n trident --use-custom-yaml
```

Instale usando o operador certificado OpenShift

Instale o Trident usando o OpenShift OperatorHub

Se você usar o Red Hat OpenShift, você pode instalar o NetApp Trident usando o operador certificado Red Hat. Use este procedimento para instalar o Trident a partir da Plataforma de contêiner do Red Hat OpenShift.

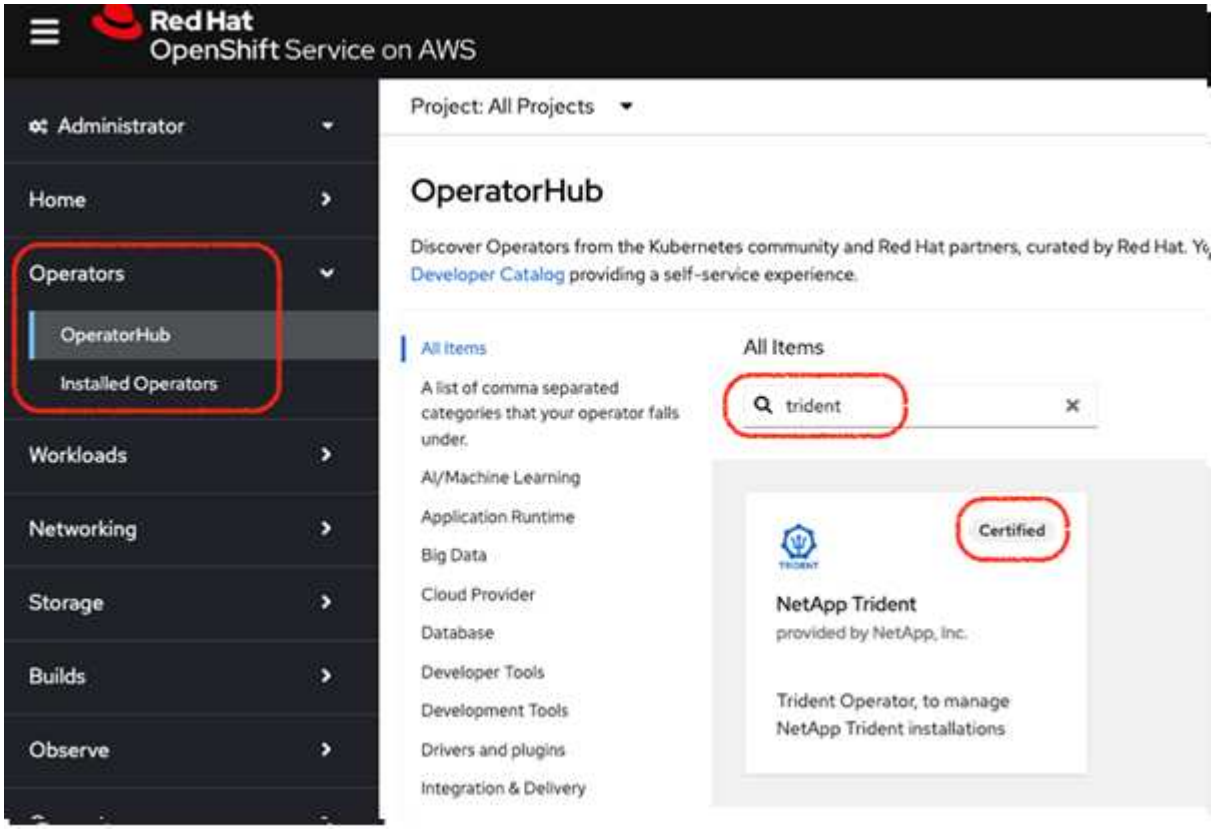
Antes de começar

Antes de iniciar a instalação, "Prepare seu ambiente para a instalação do Trident".

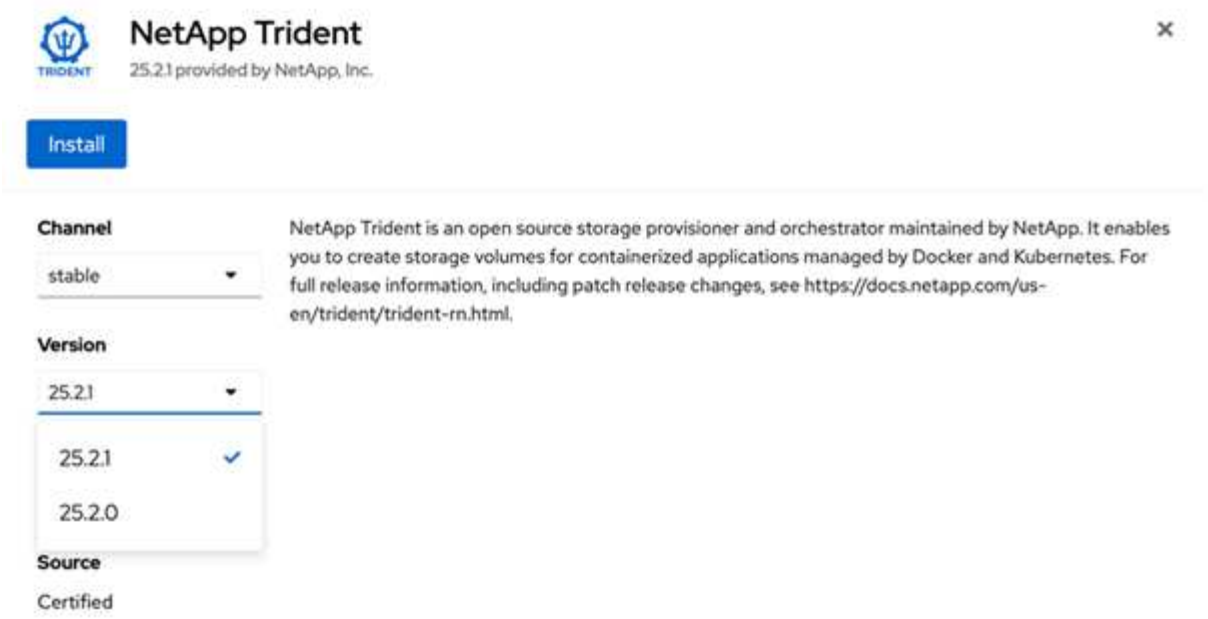
Localize e instale o operador Trident

Passos

1. Navegue até o OpenShift OperatorHub e procure por NetApp Trident.

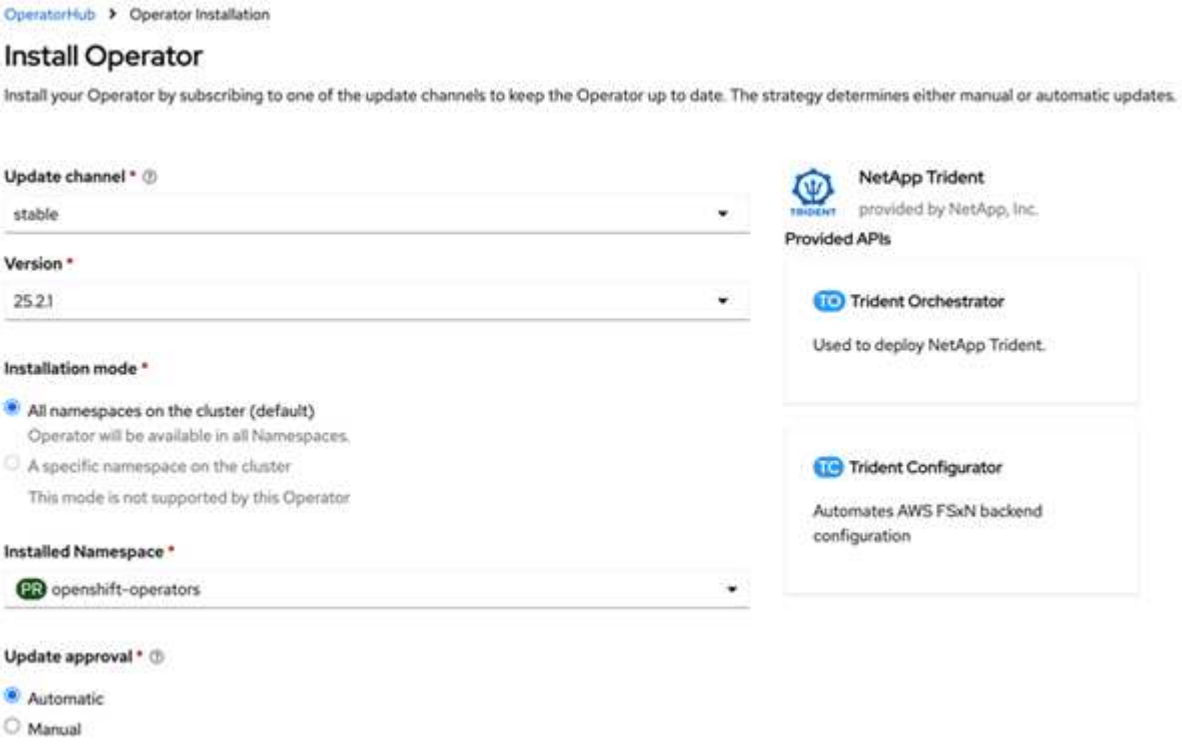


2. Clique em * NetApp Trident* para abrir as configurações de instalação.
3. Selecione as opções necessárias e clique em **Instalar** para abrir a configuração do Operador.

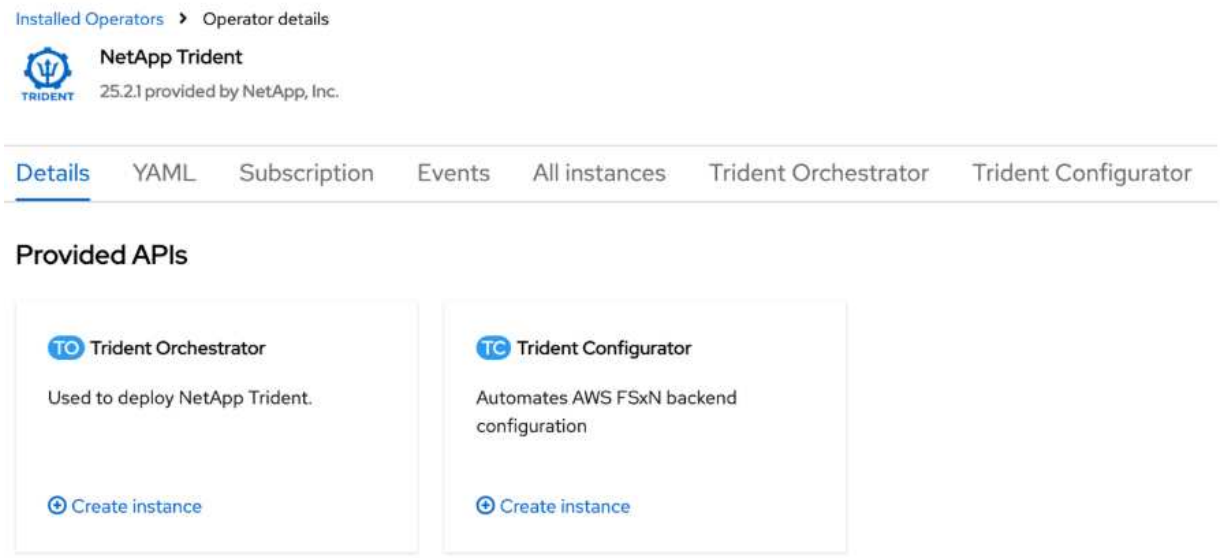


Certifique-se de selecionar a versão mais recente do Operator.

4. Mantenha todos os parâmetros como estão e clique em **Instalar**.



5. Clique em **Exibir Operador** para visualizar os detalhes do Operador.



6. Clique em **Visualização YAML** e cole o seguinte no formulário:

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
  namespace: openshift-operators
spec:
  IPv6: false
  debug: false
  nodePrep:
  - iscsi
  imageRegistry: ''
  k8sTimeout: 180s
  namespace: trident
  silenceAutosupport: false
```

A interface do usuário fornece um exemplo padrão. Você pode editá-lo diretamente em vez de copiar uma configuração completa.

Opcional: ativar concorrência

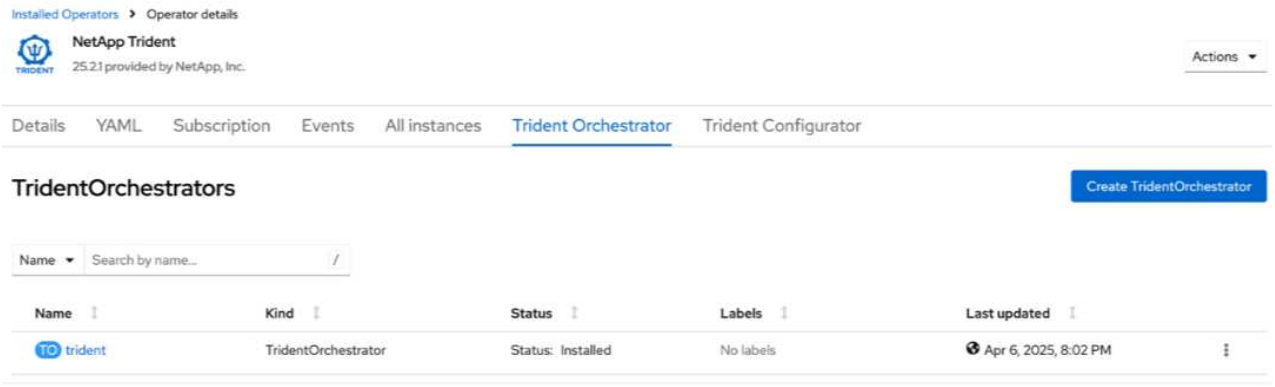
Para habilitar a concorrência, adicione o seguinte campo à especificação:

```
enableConcurrency: true
```

- Red Hat Enterprise Linux CoreOS (RHCOS) não possui iSCSI habilitado e configurado.
- Você pode adicionar o `nodePrep` parâmetro para configurar e habilitar os serviços iSCSI e Multipath em todos os nós de trabalho do OpenShift.

- A partir do OpenShift 4.19, a versão mínima do Trident suportada para esse recurso é 25.06.1.

1. Clique em **Criar**; o Trident Orchestrator será totalmente instalado.



Desinstalar operador Trident

Passos

1. Selecione o operador Trident na lista de operador instalado.
2. Selecione se pretende eliminar toda a instância do operador.

> Se você não selecionar a caixa de seleção **Excluir todas as instâncias operand deste operador**, o Trident não será desinstalado.

3. Clique em **Desinstalar**.

Mude para o operador Trident certificado OpenShift

Você pode migrar para o operador Trident certificado pela Red Hat OpenShift a partir do operador da comunidade, de uma instalação baseada em Helm ou de um operador implantado manualmente. O processo para cada método envolve desinstalar o operador existente e depois instalar o operador certificado usando o OperatorHub.

Antes de começar

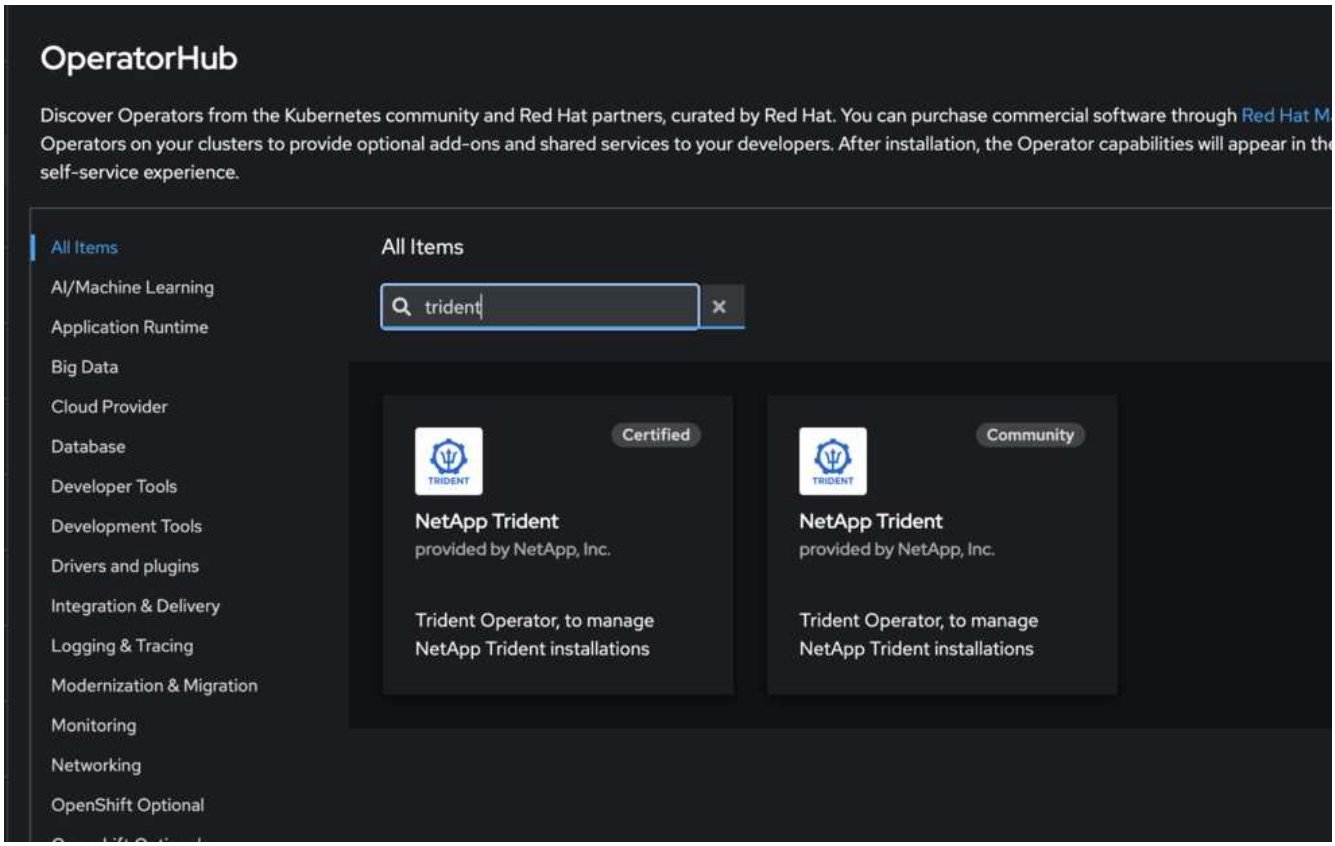
Antes de iniciar a instalação, "Prepare seu ambiente para a instalação do Trident".

> Não exclua o `TridentOrchestrator` recurso personalizado (CR) durante o processo de desinstalação. O `TridentOrchestrator` CR preserva a configuração do seu backend e da classe de armazenamento, que é necessária após a instalação do operador certificado.

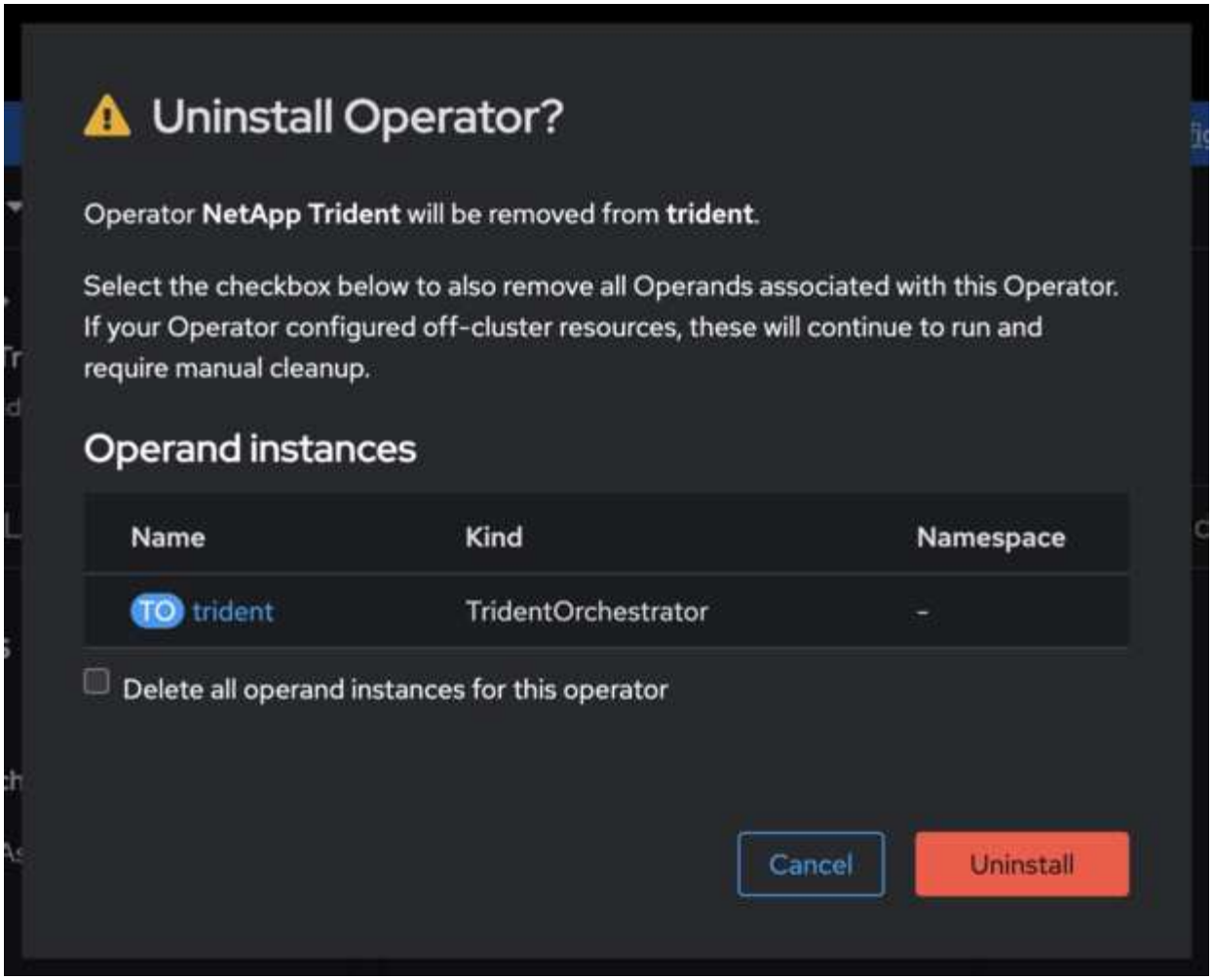
Mude do operador comunitário

Passos

1. Use o console OpenShift para navegar até o OperatorHub.



2. Encontre o operador da comunidade NetApp Trident.



Não selecione **Excluir todas as instâncias operand deste operador**.

3. Clique em **Desinstalar**.
4. Após a desinstalação ser concluída, prossiga para Instale o operador certificado OpenShift.

Migrar de uma instalação baseada em operador Helm

Passos

1. Liste o release do Helm para sua instalação do Trident:

```
helm ls -n trident
```

2. Desinstale o release do Helm:

```
helm uninstall <release-name> -n trident
```

3. Após a desinstalação ser concluída, prossiga para Instale o operador certificado OpenShift.

Mudar de um operador implantado manualmente

Se você instalou Trident implantando manualmente o operador usando um `bundle.yaml` do pacote de instalação, remova-o excluindo o mesmo manifesto.

Passos

1. Exclua a implantação do operador usando o bundle manifest:

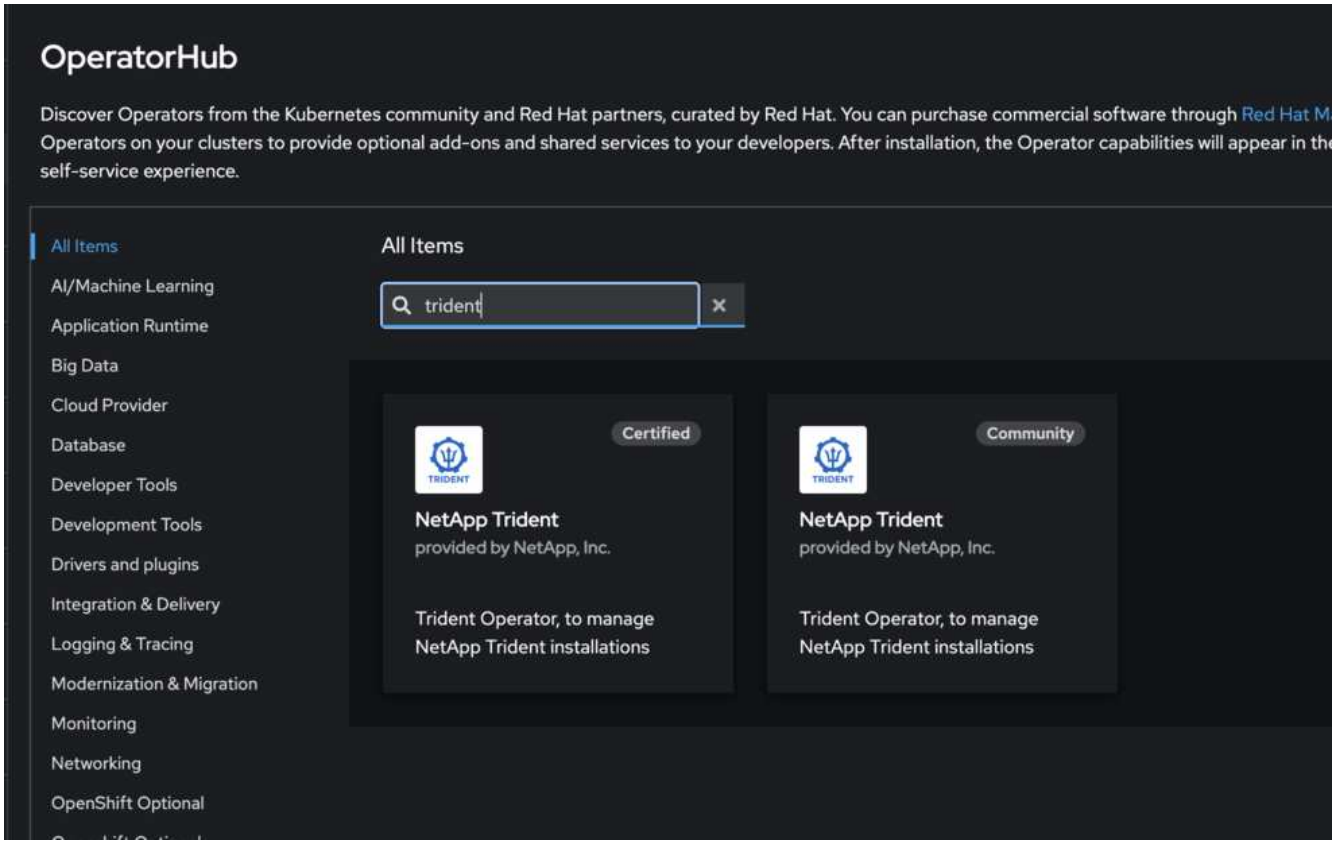
```
kubectl delete -f deploy/bundle.yaml -n trident
```

2. Após a desinstalação ser concluída, prossiga para Instale o operador certificado OpenShift.

Instale o operador certificado OpenShift

Passos

1. Navegue até o Red Hat OperatorHub.
2. Procure e selecione o Operador NetApp Trident.



3. Siga as instruções apresentadas no ecrã para instalar o operador.

Verificação

- Verifique o OperatorHub na consola para garantir que o novo operador certificado foi instalado com sucesso.

Use o Trident

Prepare o nó de trabalho do Trident

Todos os nós de trabalho no cluster do Kubernetes precisam ser capazes de montar os volumes provisionados para os pods. Para preparar os nós de trabalho, é necessário instalar ferramentas NFS, iSCSI, NVMe/TCP ou FC com base na seleção de driver.

Selecionar as ferramentas certas

Se você estiver usando uma combinação de drivers, você deve instalar todas as ferramentas necessárias para seus drivers. Versões recentes do Red Hat Enterprise Linux CoreOS (RHCOS) têm as ferramentas instaladas por padrão.

Ferramentas NFS

"Instalar as ferramentas NFS"Se você estiver usando: `ontap-nas` , `ontap-nas-economy` , `ontap-nas-flexgroup` , ou `azure-netapp-files` .

Ferramentas iSCSI

"Instale as ferramentas iSCSI" se estiver a utilizar: `ontap-san`, `ontap-san-economy`, `solidfire-san`.

Ferramentas NVMe

"Instalar as ferramentas do NVMe" Se você estiver usando `ontap-san` o protocolo NVMe (não-volátil Memory Express) em TCP (NVMe/TCP).

> A NetApp recomenda o ONTAP 9,12 ou posterior para NVMe/TCP.

Ferramentas SCSI sobre FC

```
a link:https://docs.netapp.com/us-en/ontap/san-config/configure-fc-nvme-hosts-ha-pairs-reference.html["Maneiras de configurar hosts SAN FC  FC-NVMe"]Consulte para obter mais informações sobre como configurar seus hosts SAN FC e FC-NVMe.
```

"Instalar as ferramentas FC" Se você estiver usando `ontap-san` com sanType (`fcp`SCSI sobre FC).

Pontos a considerar: * SCSI sobre FC é suportado em ambientes OpenShift e KubeVirt. * SCSI sobre FC não é suportado no Docker. * A autorrecuperação iSCSI não se aplica a SCSI via FC.

Ferramentas SMB

"Prepare-se para provisionar volumes SMB" Se você estiver usando: `ontap-nas` Para provisionar volumes SMB.

Detecção de serviço de nós

O Trident tenta detetar automaticamente se o nó pode executar serviços iSCSI ou NFS.

> A descoberta de serviço de nó identifica os serviços descobertos, mas não garante que os

serviços estejam configurados corretamente. Por outro lado, a ausência de um serviço descoberto não garante que a montagem de volume falhe.

Rever eventos

O Trident cria eventos para o nó para identificar os serviços descobertos. Para rever estes eventos, execute:

```
kubectl get event -A --field-selector involvedObject.name=<Kubernetes node name>
```

Reveja os serviços descobertos

O Trident identifica os serviços ativados para cada nó no CR do nó Trident. Para visualizar os serviços descobertos, execute:

```
tridentctl get node -o wide -n <Trident namespace>
```

Volumes NFS

Instale as ferramentas NFS usando os comandos do seu sistema operacional. Certifique-se de que o serviço NFS seja iniciado durante o tempo de inicialização.

RHEL 8 MAIS

```
sudo yum install -y nfs-utils
```

Ubuntu

```
sudo apt-get install -y nfs-common
```

Reinicie seus nós de trabalho após instalar as ferramentas NFS para evitar falhas ao anexar volumes a contêineres.

Volumes iSCSI

O Trident pode estabelecer automaticamente uma sessão iSCSI, digitalizar LUNs e descobrir dispositivos multipath, formatá-los e montá-los em um pod.

Recursos de autorrecuperação iSCSI

Para sistemas ONTAP, o Trident executa a autorrecuperação iSCSI a cada cinco minutos para:

1. **Identifique** o estado de sessão iSCSI desejado e o estado atual da sessão iSCSI.
2. **Compare** o estado desejado com o estado atual para identificar as reparações necessárias. O Trident determina as prioridades de reparação e quando efetuar as reparações.
3. **Efetuar reparações** necessárias para repor o estado atual da sessão iSCSI para o estado de sessão

iSCSI pretendido.

> Logs de atividade de auto-cura estão localizados no `trident-main` recipiente no respetivo pod Daemonset. Para visualizar registos, tem de ter definido `debug` como "verdadeiro" durante a instalação do Trident.

Os recursos de autorrecuperação iSCSI da Trident podem ajudar a impedir:

- Sessões iSCSI obsoletas ou não saudáveis que podem ocorrer após um problema de conetividade de rede. No caso de uma sessão obsoleta, o Trident aguarda sete minutos antes de sair para restabelecer a conexão com um portal.

> Por exemplo, se os segredos CHAP foram girados no controlador de armazenamento e a rede perder a conetividade, os segredos CHAP antigos (*stale*) podem persistir. A auto-cura pode reconhecer isso e restabelecer automaticamente a sessão para aplicar os segredos CHAP atualizados.

- Sessões iSCSI em falta
- LUNs em falta

Pontos a considerar antes de atualizar o Trident

- Se apenas os grupos por nó (introduzidos em mais de 23,04) estiverem em uso, a recuperação automática iSCSI iniciará os rescans SCSI para todos os dispositivos no barramento SCSI.
- Se apenas grupos com escopo de back-end (obsoletos a partir de 23,04) estiverem em uso, a recuperação automática iSCSI iniciará as reconfigurações SCSI para IDs de LUN exatas no barramento SCSI.
- Se uma combinação de grupos por nó e grupos com escopo de back-end estiver em uso, a recuperação automática iSCSI iniciará as substituições SCSI para IDs LUN exatas no barramento SCSI.

Instale as ferramentas iSCSI

Instale as ferramentas iSCSI utilizando os comandos do seu sistema operativo.

Antes de começar

- Cada nó no cluster do Kubernetes precisa ter uma IQN exclusiva. **Este é um pré-requisito necessário**.
- Se estiver usando RHCOS versão 4,5 ou posterior, ou outra distribuição Linux compatível com RHEL, com o `solidfire-san` driver e o Element OS 12,5 ou anterior, verifique se o algoritmo de autenticação CHAP está definido como MD5 em `/etc/iscsi/iscsid.conf`. algoritmos CHAP compatíveis com FIPS seguros SHA1, SHA-256 e SHA3-256 estão disponíveis com o elemento 12,7.

```
sudo sed -i 's/^\(node.session.auth.chap_algs\).*/\1 = MD5/'
/etc/iscsi/iscsid.conf
```

- Ao usar nós de trabalho que executam RHEL/Red Hat Enterprise Linux CoreOS (RHCOS) com iSCSI PVs, especifique a `discard` mountOption no StorageClass para executar a recuperação de espaço em linha. Consulte a "Documentação da Red Hat".
- Certifique-se de ter atualizado para a versão mais recente do `multipath-tools` .

RHEL 8 MAIS

1. Instale os seguintes pacotes de sistema:

```
sudo yum install -y lsscsi iscsi-initiator-utils device-mapper-
multipath
```

2. Verifique se a versão iscsi-iniciador-utils é 6,2.0,874-2.el7 ou posterior:

```
rpm -q iscsi-initiator-utils
```

3. Definir a digitalização para manual:

```
sudo sed -i 's/^\(node.session.scan\).*/\1 = manual/'
/etc/iscsi/iscsid.conf
```

4. Ativar multipathing:

```
sudo mpathconf --enable --with_multipathd y --find_multipaths n
```

> Certifique-se de `/etc/multipath.conf` que contém `find_multipaths no defaults` em .

5. Certifique-se de que `iscsid` e `multipathd` estão a funcionar:

```
sudo systemctl enable --now iscsid multipathd
```

6. Ativar e iniciar `iscsi`:

```
sudo systemctl enable --now iscsi
```

Ubuntu

1. Instale os seguintes pacotes de sistema:

```
sudo apt-get install -y open-iscsi lsscsi sg3-utils multipath-tools
scsitools
```

2. Verifique se a versão Open-iscsi é 2,0.874-5ubuntu2.10 ou posterior (para bionic) ou 2,0.874-7.1ubuntu6.1 ou posterior (para focal):

```
dpkg -l open-iscsi
```

3. Definir a digitalização para manual:

```
sudo sed -i 's/^\(node.session.scan\).*/\1 = manual/'
/etc/iscsi/iscsid.conf
```

4. Ativar multipathing:

```
sudo tee /etc/multipath.conf <<-EOF
defaults {
    user_friendly_names yes
    find_multipaths no
}
EOF
sudo systemctl enable --now multipath-tools.service
sudo service multipath-tools restart
```

> Certifique-se de `/etc/multipath.conf` que contém `find_multipaths no` `defaults` em .

5. Certifique-se de que `open-iscsi` e `multipath-tools` estão ativados e em execução:

```
sudo systemctl status multipath-tools
sudo systemctl enable --now open-iscsi.service
sudo systemctl status open-iscsi
```

> Para o Ubuntu 18,04, você deve descobrir portas de destino com `iscsiadm` antes de iniciar `open-iscsi` o daemon iSCSI para iniciar. Em alternativa, pode modificar o `iscsi` serviço para iniciar `iscsid` automaticamente.

Configurar ou desativar a auto-recuperação iSCSI

Você pode configurar as seguintes configurações de auto-recuperação iSCSI Trident para corrigir sessões obsoletas:

- **Intervalo de auto-recuperação iSCSI**: Determina a frequência na qual a auto-recuperação iSCSI é invocada (predefinição: 5 minutos). Você pode configurá-lo para executar com mais frequência definindo um número menor ou com menos frequência definindo um número maior.

> Definir o intervalo de auto-recuperação iSCSI para 0 interrompe completamente a auto-recuperação iSCSI. Não recomendamos a desativação do iSCSI Self-healing; ele só deve ser desativado em certos cenários quando o iSCSI Self-healing não estiver funcionando como

pretendido ou para fins de depuração.

- **Tempo de espera de auto-cura iSCSI**: Determina a duração de espera de auto-recuperação iSCSI antes de sair de uma sessão não saudável e tentar iniciar sessão novamente (predefinição: 7 minutos). Você pode configurá-lo para um número maior para que as sessões identificadas como não saudáveis tenham que esperar mais antes de serem desconetadas e, em seguida, uma tentativa é feita para fazer login novamente, ou um número menor para fazer logout e fazer login mais cedo.

Leme

Para configurar ou alterar as definições de recuperação automática iSCSI, passe os `iscsiSelfHealingInterval` parâmetros e `iscsiSelfHealingWaitTime` durante a instalação do leme ou atualização do leme.

O exemplo a seguir define o intervalo de auto-recuperação iSCSI para 3 minutos e o tempo de espera de auto-recuperação para 6 minutos:

```
helm install trident trident-operator-100.2610.0.tgz --set
iscsiSelfHealingInterval=3m0s --set iscsiSelfHealingWaitTime=6m0s -n
trident
```

tridentctl

Para configurar ou alterar as configurações de auto-recuperação iSCSI, passe os `iscsi-self-healing-interval` parâmetros e `iscsi-self-healing-wait-time` durante a instalação ou atualização do tridentctl.

O exemplo a seguir define o intervalo de auto-recuperação iSCSI para 3 minutos e o tempo de espera de auto-recuperação para 6 minutos:

```
tridentctl install --iscsi-self-healing-interval=3m0s --iscsi-self
-healing-wait-time=6m0s -n trident
```

Volumes NVMe/TCP

Instale as ferramentas NVMe usando os comandos do seu sistema operacional.

- O NVMe requer o RHEL 9 ou posterior.
- Se a versão do kernel do seu nó Kubernetes for muito antiga ou se o pacote NVMe não estiver disponível para a versão do kernel, talvez seja necessário atualizar a versão do kernel do nó para uma com o pacote NVMe.

RHEL 9

```
sudo yum install nvme-cli
sudo yum install linux-modules-extra-$(uname -r)
```

```
sudo modprobe nvme-tcp
```

Ubuntu

```
sudo apt install nvme-cli
sudo apt -y install linux-modules-extra-$(uname -r)
sudo modprobe nvme-tcp
```

Verifique a instalação

Após a instalação, verifique se cada nó no cluster do Kubernetes tem um NQN exclusivo usando o comando:

```
cat /etc/nvme/hostnqn
```

> O Trident modifica o `ctrl_device_tmo` valor para garantir que o NVMe não desista do caminho se ele cair. Não altere esta definição.

SCSI em volumes FC

Agora você pode usar o protocolo Fibre Channel (FC) com o Trident para provisionar e gerenciar recursos de storage no sistema ONTAP.

Pré-requisitos

Configure as configurações de rede e nó necessárias para FC.

Definições de rede

1. Obtenha o WWPN das interfaces de destino. "mostra da interface de rede"Consulte para obter mais informações.
2. Obtenha o WWPN para as interfaces no iniciador (Host).

   Consulte os utilitários do sistema operacional host correspondentes.

3. Configure o zoneamento no switch FC usando WWPNs do host e do destino.

   Consulte a documentação do fornecedor do switch responsável para obter informações.

   Consulte a seguinte documentação do ONTAP para obter detalhes:

   - "Visão geral do zoneamento Fibre Channel e FCoE"
   - "Maneiras de configurar hosts SAN FC FC-NVMe"

Instalar as ferramentas FC

Instale as ferramentas FC usando os comandos do seu sistema operacional.

- Ao usar nós de trabalho que executam RHEL/Red Hat Enterprise Linux CoreOS (RHCOS) com FC PVs,

especifique a `discard` mountOption no StorageClass para executar a recuperação de espaço em linha. Consulte a "Documentação da Red Hat".

RHEL 8 MAIS

1. Instale os seguintes pacotes de sistema:

```
sudo yum install -y lsscsi device-mapper-multipath
```

2. Ativar multipathing:

```
sudo mpathconf --enable --with_multipathd y --find_multipaths n
```

> Certifique-se de `/etc/multipath.conf` que contém `find_multipaths no defaults` em .

3. Certifique-se de que `multipathd` está em execução:

```
sudo systemctl enable --now multipathd
```

Ubuntu

1. Instale os seguintes pacotes de sistema:

```
sudo apt-get install -y lsscsi sg3-utils multipath-tools scsitools
```

2. Ativar multipathing:

```
sudo tee /etc/multipath.conf <<-EOF
defaults {
    user_friendly_names yes
    find_multipaths no
}
EOF
sudo systemctl enable --now multipath-tools.service
sudo service multipath-tools restart
```

> Certifique-se de `/etc/multipath.conf` que contém `find_multipaths no defaults` em .

3. Certifique-se de que `multipath-tools` está ativado e em execução:

```
sudo systemctl status multipath-tools
```

Prepare-se para provisionar volumes SMB

Você pode provisionar volumes SMB usando `ontap-nas` motoristas.

> É necessário configurar os protocolos NFS e SMB/CIFS na SVM para criar um `ontap-nas-economy` volume SMB para clusters no local do ONTAP. A falha na configuração desses protocolos fará com que a criação de volume SMB falhe.

> `autoExportPolicy` Não é compatível com volumes SMB.

Antes de começar

Antes de provisionar volumes SMB, você deve ter o seguinte:

- Um cluster do Kubernetes com um nó de controlador Linux e pelo menos um nó de trabalho do Windows que executa o Windows Server 2022. O Trident dá suporte a volumes SMB montados em pods executados apenas em nós do Windows.
- Pelo menos um segredo do Trident contendo suas credenciais do ative Directory. Para gerar segredo `smbcreds`:

```
kubectl create secret generic smbcreds --from-literal username=user
--from-literal password='password'
```

- Um proxy CSI configurado como um serviço Windows. Para configurar um `csi-proxy`, "GitHub: CSI Proxy"consulte ou "GitHub: CSI Proxy para Windows" para nós do Kubernetes executados no Windows.

Passos

1. Para o ONTAP no local, você pode criar, opcionalmente, um compartilhamento SMB ou o Trident pode criar um para você.

   > Compartilhamentos SMB são necessários para o Amazon FSX for ONTAP.

   Você pode criar os compartilhamentos de administração SMB de duas maneiras usando o "Microsoft Management Console"snap-in pastas compartilhadas ou usando a CLI do ONTAP. Para criar compartilhamentos SMB usando a CLI do ONTAP:

   a. Se necessário, crie a estrutura do caminho do diretório para o compartilhamento.

      O `vserver cifs share create` comando verifica o caminho especificado na opção -path durante a criação de compartilhamento. Se o caminho especificado não existir, o comando falhará.

   b. Crie um compartilhamento SMB associado ao SVM especificado:

```
vserver cifs share create -vserver vserver_name -share-name
```

```
share_name -path path [-share-properties share_properties,...]
[other_attributes] [-comment text]
```

c. Verifique se o compartilhamento foi criado:

```
vserver cifs share show -share-name share_name
```

"Crie um compartilhamento SMB"Consulte para obter detalhes completos.

2. Ao criar o back-end, você deve configurar o seguinte para especificar volumes SMB. Para obter todas as opções de configuração de back-end do FSX for ONTAP, "Opções e exemplos de configuração do FSX for ONTAP"consulte .

Parâmetro	Descrição	Exemplo
`smbShare`	Você pode especificar uma das seguintes opções: O nome de um compartilhamento SMB criado usando o Console de Gerenciamento da Microsoft ou a CLI do ONTAP; um nome para permitir que o Trident crie o compartilhamento SMB; ou você pode deixar o parâmetro em branco para impedir o acesso comum ao compartilhamento a volumes. Esse parâmetro é opcional para o ONTAP no local. Esse parâmetro é necessário para backends do Amazon FSX for ONTAP e não pode ficar em branco.	`smb-share`
`nasType`	**Tem de estar definido para `smb`.** Se nulo, o padrão é `nfs`.	`smb`
`securityStyle`	Estilo de segurança para novos volumes. **Deve ser definido como `ntfs` ou `mixed` para volumes SMB.**	`ntfs` Ou `mixed` para volumes SMB
`unixPermissions`	Modo para novos volumes. **Deve ser deixado vazio para volumes SMB.**	""

Provisionar storage de objetos com COSI

Use a Container Object Storage Interface (COSI) para provisionar buckets S3 para cargas de trabalho do Kubernetes por meio de recursos do Kubernetes gerenciados pelo Trident.

Visão geral

O COSI conecta provedores de storage de objetos a cargas de trabalho do Kubernetes. O Trident fornece os seguintes recursos:

- `TridentObjectStore`: conecta o Trident a um sistema de storage de objetos e às suas credenciais. Crie este recurso no namespace do Trident.

- `BucketClass`: define a seleção do armazenamento de objetos e a política de exclusão de buckets.
- `BucketClaim`: solicita um bucket.
- `BucketAccessClass`: define como o acesso aos buckets é provisionado.
- `BucketAccess`: solicita credenciais para um ou mais buckets.

O Trident oferece suporte ao COSI com os seguintes drivers de storage de objetos:

- O StorageGRID: geralmente disponível.
- O ONTAP S3: prévia técnica.

O Trident suporta o protocolo S3. Na versão inicial, há suporte apenas ao endereçamento S3 no estilo de caminho.

Requisitos e limitações

- Habilite ambos `enableConcurrency` e `enableCOSI` no `TridentOrchestrator` recurso personalizado (TORC). O COSI não está habilitado por padrão.
- Antes de habilitar a concorrência do controlador, verifique se todos os backends do Trident configurados usam drivers compatíveis. Um driver de backend incompatível impede o Trident de iniciar; consulte "Antes de ativar a escalabilidade da controladora".
- Não execute o Trident COSI juntamente com o controlador COSI da comunidade.
- A versão inicial utiliza a `objectstorage.k8s.io/v1alpha2` API COSI.
- Os recursos do StorageGRID `TridentObjectStore` usam credenciais raiz de locatário de longa duração. Cada `TridentObjectStore` recurso se conecta a um locatário do StorageGRID; vários recursos podem se conectar ao mesmo locatário. Armazene as credenciais em um Secret do Kubernetes no namespace Trident e proteja-as de acordo com seus requisitos de segurança.
- O StorageGRID `BucketAccess` suporta até oito buckets.

Para informações gerais sobre operações de controladores concorrentes, consulte "Escalabilidade do controlador".

Habilitar COSI

Defina as duas opções na `TridentOrchestrator` especificação:

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  namespace: trident
  enableConcurrency: true
  enableCOSI: true
```

Aplique o TORC ou atualize seu recurso existente usando seu fluxo de trabalho operacional normal. Para obter mais informações sobre como configurar o TORC, consulte "Personalizar a instalação do operador

Trident".

Configure um armazenamento de objetos do StorageGRID

Crie um Secret do Kubernetes com as credenciais de tenant do StorageGRID no namespace do Trident:

```
apiVersion: v1
kind: Secret
metadata:
  name: tenant3
  namespace: trident
type: Opaque
stringData:
  username: <tenant-root-username>
  password: <tenant-root-password>
```

Crie um `TridentObjectStore` objeto que faça referência ao Secret. Substitua os endpoints de exemplo e o ID da conta pelos valores do seu ambiente StorageGRID:

```
apiVersion: trident.netapp.io/v1
kind: TridentObjectStore
metadata:
  name: trident-tenant3
  namespace: trident
spec:
  accountId: "<account-id>"
  credentials:
    secretName: tenant3
  labels:
    tenantInstance: three
  managementEndpoint: https://<storagegrid-management-endpoint>
  objectStoreDriver: storage-grid
  s3Endpoint: https://<storagegrid-s3-endpoint>
```

Aplique ambos os recursos:

```
kubectl apply -f tenant-secret.yaml
kubectl apply -f trident-object-store.yaml
```

Configurar um armazenamento de objetos ONTAP S3



O suporte ao ONTAP S3 por meio do COSI é uma prévia técnica no Trident 26.10 e não é compatível com uso em produção. Os campos de configuração e o comportamento podem mudar antes da disponibilidade geral.

Crie um Secret do Kubernetes com as credenciais de usuário do ONTAP S3 no namespace do Trident e, em seguida, crie um `TridentObjectStore` que faça referência ao Secret. Substitua os placeholders pelos valores do seu ambiente ONTAP S3:

```
apiVersion: v1
kind: Secret
metadata:
  name: ontap-s3-credentials
  namespace: trident
type: Opaque
stringData:
  username: "<user>"
  password: "<password>"
---
apiVersion: trident.netapp.io/v1
kind: TridentObjectStore
metadata:
  name: ontap-s3
  namespace: trident
spec:
  objectStoreDriver: ontap-s3
  objectStoreName: ontap-object-store
  managementEndpoint: "<management-endpoint>"
  svmName: "<svm-name>"
  storagePrefix: trident
  credentials:
    secretName: ontap-s3-credentials
```

O `storagePrefix` é opcional e o padrão é `trident` quando omitido. Aplique o Secret e `TridentObjectStore`:

```
kubectl apply -f ontap-s3-object-store.yaml
```

Crie uma BucketClass e solicite um bucket

Um `BucketClass` pode selecionar um armazenamento de objetos usando um seletor de rótulo, um tipo de armazenamento de objetos ou ambos. Sem parâmetros de seleção, o Trident pode selecionar entre todos os armazenamentos de objetos configurados.

```
apiVersion: objectstorage.k8s.io/v1alpha2
kind: BucketClass
metadata:
  name: sg-delete
spec:
  driverName: cosi.trident.netapp.io
```

```
  deletionPolicy: Delete
  parameters:
    selector: tenantInstance=three
    objectStoreType: storage-grid
```

Crie um `BucketClaim` que utilize a classe:

```
apiVersion: objectstorage.k8s.io/v1alpha2
kind: BucketClaim
metadata:
  name: claim-a1
  namespace: default
spec:
  bucketClassName: sg-delete
  protocols:
    - S3
```

Quando o Trident processa uma solicitação, ele seleciona um armazenamento de objetos correspondente. Se vários armazenamentos de objetos corresponderem, o Trident escolherá um aleatoriamente para distribuir os buckets entre os armazenamentos de objetos correspondentes.

Conceder acesso de carga de trabalho a um bucket

Crie um `BucketAccessClass` para definir o provisionamento de acesso:

```
apiVersion: objectstorage.k8s.io/v1alpha2
kind: BucketAccessClass
metadata:
  name: sg-access
spec:
  authenticationType: Key
  driverName: cosi.trident.netapp.io
  multiBucketAccess: MultipleBuckets
```

Crie um `BucketAccess` que faça referência à declaração do bucket:

```
apiVersion: objectstorage.k8s.io/v1alpha2
kind: BucketAccess
metadata:
  name: sg-claim-a1
  namespace: default
spec:
  bucketAccessClassName: sg-access
  bucketClaims:
```

```
    - accessMode: ReadWrite
      accessSecretName: sg-claim-a1
      bucketClaimName: claim-a1
  protocol: S3
```

O Trident cria um Secret do Kubernetes contendo as informações de acesso ao bucket. Uma carga de trabalho pode consumir esse Secret usando mecanismos padrão de Secret do Kubernetes. O Secret gerado inclui detalhes de conexão do S3, como o identificador do bucket, o endpoint, a região e as chaves de acesso. Não exponha o Secret gerado a usuários ou cargas de trabalho não autorizados.

O segredo gerado inclui chaves como `COSI_S3_ACCESS_KEY_ID`, `COSI_S3_ACCESS_SECRET_KEY`, `COSI_S3_ADDRESSING_STYLE`, `COSI_S3_BUCKET_ID`, `COSI_S3_ENDPOINT` e `COSI_S3_REGION`.

Configurar e gerenciar backends

Configure os backends do Trident

Um back-end define a relação entre o Trident e um sistema de storage. Ele informa à Trident como se comunicar com esse sistema de storage e como o Trident deve provisionar volumes a partir dele.

O Trident oferece automaticamente pools de storage de back-ends que atendem aos requisitos definidos por uma classe de storage. Saiba como configurar o back-end para o seu sistema de armazenamento.

- "Configurar um back-end do Azure NetApp Files"
- "Configurar um back-end do Google Cloud NetApp volumes"
- "Configurar um back-end NetApp HCI ou SolidFire"
- "Configurar um back-end com drivers nas ONTAP ou Cloud Volumes ONTAP"
- "Configure um back-end com drivers SAN ONTAP ou Cloud Volumes ONTAP"
- "Use o Trident com o Amazon FSX para NetApp ONTAP"

Azure NetApp Files

Configurar um back-end do Azure NetApp Files

Use o Azure NetApp Files como backend para Trident. Esse backend oferece suporte a volumes NFS e SMB. Trident oferece suporte a identidades gerenciadas e identidade de carga de trabalho para clusters do Azure Kubernetes Service (AKS).

Ambientes de nuvem Azure suportados

Trident oferece suporte a back-ends do Azure NetApp Files em vários ambientes de nuvem do Azure.

As nuvens do Microsoft Azure compatíveis incluem:

- Azure Commercial
- Azure Government (Azure Government / MAG)

- O Azure China

Ao implantar Trident ou configurar um backend do Azure NetApp Files, certifique-se de que o Azure Resource Manager e os endpoints de autenticação correspondam ao seu ambiente de nuvem do Azure.

Revise o suporte do driver do Azure NetApp Files

Trident fornece o seguinte driver de armazenamento Azure NetApp Files.

Os modos de acesso suportados incluem *ReadWriteOnce* (RWO), *ReadOnlyMany* (ROX), *ReadWriteMany* (RWX) e *ReadWriteOncePod* (RWOP).

Condutor	Protocolo	VolumeMode	Modos de acesso suportados	Sistemas de arquivos suportados
`azure-netapp-files`	NFS, SMB	Sistema de ficheiros	RWO, ROX, RWX, RWOP	`nfs`, `smb`

Revisar considerações

- Azure NetApp Files não suporta volumes menores que 50 GiB. Trident cria um volume de 50 GiB quando um volume menor é solicitado.
- O Trident dá suporte a volumes SMB montados em pods executados apenas em nós do Windows.
- As implantações do Azure NetApp Files em nuvens Azure não comerciais exigem endpoints do Azure Resource Manager e de autenticação específicos da nuvem. Certifique-se de que Trident e qualquer configuração de backend usem os endpoints apropriados para o seu ambiente de nuvem Azure.

Use identidades gerenciadas para AKS

Trident oferece suporte "identidades gerenciadas" para clusters AKS.

Se você utiliza `tridentctl` para criar ou gerenciar back-ends do Azure NetApp Files, certifique-se de que ele esteja configurado para o ambiente de nuvem Azure correto.

Para usar identidades gerenciadas, você deve ter:

- Um cluster do Kubernetes implantado usando AKS
- Identidades gerenciadas configuradas no cluster Kubernetes do AKS
- Trident instalado com `cloudProvider` definido para `"Azure"`

Operador Trident

Edite `tridentorchestrator_cr.yaml` e defina `cloudProvider` para `"Azure"`.

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
```

```
  imagePullPolicy: IfNotPresent
  cloudProvider: "Azure"
```

Leme

O exemplo a seguir instala Trident e define `cloudProvider` usando a variável de ambiente `$CP`:

```
helm install trident trident-operator-100.2610.0.tgz --create-namespace
--namespace <trident-namespace> --set cloudProvider=$CP
```

<code>dtridentctl</code>

O exemplo a seguir instala Trident e define o `cloud-provider` sinalizador para `Azure`:

```
tridentctl install --cloud-provider="Azure" -n trident
```

Use identidade de carga de trabalho para AKS

A identidade de carga de trabalho permite que os pods do Kubernetes acessem recursos do Azure autenticando-se como uma identidade de carga de trabalho.

Se você utiliza `tridentctl` para criar ou gerenciar back-ends do Azure NetApp Files, certifique-se de que ele esteja configurado para o ambiente de nuvem Azure correto.

Para usar workload identity, você precisa ter:

- Um cluster do Kubernetes implantado usando AKS
- Identidade da carga de trabalho e oidc-emitente configurados no cluster AKS Kubernetes
- Trident instalado com `cloudProvider` definido para `"Azure"` e `cloudIdentity` definido para o valor de identidade da carga de trabalho

Operador Trident

Edite `tridentorchestrator_cr.yaml` e defina `cloudProvider` como `"Azure"`. Defina `cloudIdentity` como `azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxx`.

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  imagePullPolicy: IfNotPresent
  cloudProvider: "Azure"
```

```
  cloudIdentity: 'azure.workload.identity/client-id: xxxxxxxx-xxxx-
xxxx-xxxx-xxxxxxxxxxx' # Edit
```

Leme

Defina os valores para os parâmetros **cloud-provider (CP)** e **cloud-identity (CI)** usando as seguintes variáveis de ambiente:

```
export CP="Azure"
export CI="'azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxxx'"
```

O exemplo a seguir instala Trident e define `cloudProvider` usando `$CP` e define `cloudIdentity` usando `$CI`:

```
helm install trident trident-operator-100.6.0.tgz --set
cloudProvider=$CP --set cloudIdentity="$CI"
```

<code>dtridentctl</code>

Defina os valores para os parâmetros **cloud provider** e **cloud identity** usando as seguintes variáveis de ambiente:

```
export CP="Azure"
export CI="azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxxx"
```

O exemplo a seguir instala Trident e define `cloud-provider` como `$CP` e `cloud-identity` como `$CI`:

```
tridentctl install --cloud-provider=$CP --cloud-identity="$CI" -n
trident
```

Prepare-se para configurar um back-end do Azure NetApp Files

Antes de configurar o back-end do Azure NetApp Files, você precisa garantir que os seguintes requisitos sejam atendidos.

Ambientes de nuvem Azure suportados

Trident oferece suporte a back-ends do Azure NetApp Files em vários ambientes de nuvem do Azure.

As nuvens do Microsoft Azure compatíveis incluem:

- Azure Commercial
- Azure Government (Azure Government / MAG)
- O Azure China

Ao preparar seu ambiente, certifique-se de que sua assinatura do Azure, configuração de identidade e recursos do Azure NetApp Files sejam criados no ambiente de nuvem do Azure apropriado.

Pré-requisitos para volumes NFS e SMB

Se você estiver usando o Azure NetApp Files pela primeira vez ou em um novo local, será necessária alguma configuração inicial para configurar o Azure NetApp Files e criar um volume NFS. Consulte "Azure: Configure o Azure NetApp Files e crie um volume NFS".

Para configurar e usar um "Azure NetApp Files" backend, você precisa do seguinte:

> - `subscriptionID` `tenantID`, , `clientID`, , `location` E `clientSecret` são opcionais ao usar identidades gerenciadas em um cluster AKS.
> - `tenantID` `clientID`, , E `clientSecret` são opcionais ao usar uma identidade de nuvem em um cluster AKS.
> - As implantações do Azure NetApp Files em nuvens Azure não comerciais exigem endpoints do Azure Resource Manager e de autenticação específicos da nuvem. Certifique-se de que Trident e qualquer configuração de backend usem os endpoints apropriados para o seu ambiente de nuvem Azure.

- Um pool de capacidade. "Microsoft: Crie um pool de capacidade para o Azure NetApp Files"Consulte a .
- Uma sub-rede delegada ao Azure NetApp Files. "Microsoft: Delegar uma sub-rede ao Azure NetApp Files"Consulte a .
- `subscriptionID` A partir de uma subscrição do Azure com o Azure NetApp Files ativado.
- `tenantID`, `clientID` E `clientSecret` de um "Registo da aplicação" no Azure ative Directory com permissões suficientes para o serviço Azure NetApp Files. O Registro de aplicativos deve usar:
  - A função proprietário ou Colaborador "Pré-definido pelo Azure".
  - A "Função de Colaborador personalizada"no nível da subscrição (`assignableScopes`) com as seguintes permissões que estão limitadas apenas ao que o Trident requer. Depois de criar a função personalizada"Atribua a função usando o portal do Azure", .

    ▼ Função de colaborador personalizada

```
{
  "id": "/subscriptions/<subscription-
id>/providers/Microsoft.Authorization/roleDefinitions/<role-
definition-id>",
  "properties": {
    "roleName": "custom-role-with-limited-perms",
    "description": "custom role providing limited permissions",
    "assignableScopes": [
      "/subscriptions/<subscription-id>"
    ],
    "permissions": [
      {
        "actions": [
          "Microsoft.NetApp/netAppAccounts/capacityPools/read",
          "Microsoft.NetApp/netAppAccounts/capacityPools/write",
```

```
"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/read",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/write",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/delete",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/snapshots/re
ad",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/snapshots/wr
ite",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/snapshots/de
lete",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/MountTargets
/read",
          "Microsoft.Network/virtualNetworks/read",
          "Microsoft.Network/virtualNetworks/subnets/read",

"Microsoft.Features/featureProviders/subscriptionFeatureRegistratio
ns/read",

"Microsoft.Features/featureProviders/subscriptionFeatureRegistratio
ns/write",

"Microsoft.Features/featureProviders/subscriptionFeatureRegistratio
ns/delete",
          "Microsoft.Features/features/read",
          "Microsoft.Features/operations/read",
          "Microsoft.Features/providers/features/read",
          "Microsoft.Features/providers/features/register/action",

"Microsoft.Features/providers/features/unregister/action",

"Microsoft.Features/subscriptionFeatureRegistrations/read"
        ],
        "notActions": [],
        "dataActions": [],
        "notDataActions": []
      }
    ]
  }
}
```

- O Azure `location` que contém pelo menos um "sub-rede delegada". O `location` parâmetro é obrigatório no nível superior do arquivo de configuração de backend. Os valores de localização especificados em pools virtuais são ignorados.
- Para usar `Cloud Identity`o , obtenha o `client ID` de a "identidade gerenciada atribuída pelo usuário" e especifique esse ID no `azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxx`.

Requisitos adicionais para volumes SMB

Para criar um volume SMB, você deve ter:

- Ative Directory configurado e conetado ao Azure NetApp Files. "Microsoft: Crie e gerencie conexões do ative Directory para Azure NetApp Files"Consulte a .
- Um cluster do Kubernetes com um nó de controlador Linux e pelo menos um nó de trabalho do Windows que executa o Windows Server 2022. O Trident dá suporte a volumes SMB montados em pods executados apenas em nós do Windows.
- Pelo menos um segredo do Trident contendo suas credenciais do ative Directory para que o Azure NetApp Files possa se autenticar no ative Directory. Para gerar segredo `smbcreds`:

```
kubectl create secret generic smbcreds --from-literal username=user
--from-literal password='password'
```

- Um proxy CSI configurado como um serviço Windows. Para configurar um `csi-proxy`, "GitHub: CSI Proxy"consulte ou "GitHub: CSI Proxy para Windows" para nós do Kubernetes executados no Windows.

Exemplos e opções de configuração de back-end do Azure NetApp Files

Saiba mais sobre as opções de configuração de back-end NFS e SMB para Azure NetApp Files e reveja exemplos de configuração.

Opções de configuração de back-end

Trident usa sua configuração de back-end (sub-rede, rede virtual, nível de serviço e localização) para criar volumes do Azure NetApp Files em pools de capacidade disponíveis na localização solicitada e que correspondam ao nível de serviço e à sub-rede solicitados.

Os backends Azure NetApp Files fornecem essas opções de configuração.

Parâmetro	Descrição	Padrão
`version`	Versão de configuração do backend.	Sempre 1
`storageDriverName`	Nome do controlador de armazenamento	"ficheiros azure-NetApp"
`backendName`	Nome personalizado para o backend de armazenamento	Nome do condutor e carateres aleatórios

Parâmetro	Descrição	Padrão
`subscriptionID`	O ID de assinatura da sua assinatura do Azure Opcional quando identidades gerenciadas está habilitado em um cluster AKS.	
`tenantID`	O ID do locatário de um Registo de aplicações Opcional quando identidades geridas ou identidade na nuvem são utilizadas num cluster AKS.	
`clientID`	A ID do cliente de um registo de aplicações opcional quando identidades geridas ou identidade na nuvem são utilizadas num cluster AKS.	
`clientSecret`	O segredo do cliente de um Registo de aplicações Opcional quando identidades geridas ou identidade na nuvem são utilizadas num cluster AKS.	
`serviceLevel`	Um de `Standard`, `Premium`, ou `Ultra`	"" (aleatório)
`location`	Nome do local do Azure onde os novos volumes serão criados Opcional quando identidades gerenciadas estiverem ativadas em um cluster AKS.	
`cloudConfiguration.cloudName`	Identifica o ambiente de nuvem do Azure para o backend. Defina como `AzureGovernment` ao usar o Azure Government.	
`resourceGroups`	Lista de grupos de recursos para filtragem de recursos descobertos	"[]" (sem filtro)
`netappAccounts`	Lista de contas do NetApp para filtragem de recursos descobertos	"[]" (sem filtro)
`capacityPools`	Lista de pools de capacidade para filtrar recursos descobertos	"[]" (sem filtro, aleatório)
`virtualNetwork`	Nome de uma rede virtual com uma sub-rede delegada	""
`subnet`	Nome de uma sub-rede delegada `Microsoft.Netapp/volumes`	""

Parâmetro	Descrição	Padrão
`networkFeatures`	Conjunto de recursos de VNet para um volume, pode ser `Basic` ou `Standard`. Network Features não está disponível em todas as regiões e pode precisar ser habilitado em uma assinatura. Especificar `networkFeatures` quando a funcionalidade não está habilitada faz com que o provisionamento do volume falhe.	""
`nfsMountOptions`	Controle preciso das opções de montagem NFS. Ignorado para volumes SMB. Para montar volumes usando a versão de NFS 4.1, inclua `nfsvers=4` na lista de opções de montagem separadas por vírgula para escolher NFS v4.1. As opções de montagem definidas em uma definição de classe de armazenamento substituem as opções de montagem definidas na configuração do backend.	"3"
`limitVolumeSize`	Falha no provisionamento se o tamanho do volume solicitado estiver acima desse valor	"" (não aplicado por padrão)
`debugTraceFlags`	Debug flags para usar ao solucionar problemas. Exemplo, `\{"api": false, "method": true, "discovery": true}`. Não use isso a menos que você esteja solucionando problemas e exija um despejo de log detalhado.	nulo
`nasType`	Configurar a criação de volumes NFS ou SMB. As opções são `nfs`, `smb` ou null. A configuração como null padrão para volumes NFS.	`nfs`
`supportedTopologies`	Representa uma lista de regiões e zonas que são suportadas por este backend. Para obter mais informações, "Use a topologia CSI"consulte .	
`qosType`	Representa o tipo de QoS: Automático ou Manual.	Auto
`maxThroughput`	Define a taxa de transferência máxima permitida em MiB/s. Suportado apenas para pools de capacidade de QoS manual.	`4 MiB/sec`

Para obter mais informações sobre recursos de rede, "Configurar recursos de rede para um volume Azure NetApp Files"consulte .

Considere ambientes de nuvem Azure (26.06)

A partir da versão 26.06, Trident oferece suporte à criação e ao gerenciamento de back-ends do Azure NetApp Files em vários ambientes de nuvem do Azure.

As nuvens do Microsoft Azure compatíveis incluem:

- Azure Commercial
- Azure Government (Azure Government / MAG)
- O Azure China

Ao implantar Trident ou criar um backend do Azure NetApp Files, certifique-se de que os endpoints do Azure Resource Manager e de autenticação correspondam ao seu ambiente de nuvem do Azure. Se os endpoints não corresponderem, `tridentctl` não será possível autenticar e a criação do backend falhará.

Para um backend do Azure Government, defina `cloudConfiguration.cloudName` na configuração do backend:

```
cloudConfiguration:
  cloudName: AzureGovernment
```

Permissões e recursos necessários

Se você receber um erro "Nenhum pool de capacidade encontrado" ao criar um PVC, é provável que o registro do seu aplicativo não tenha as permissões e os recursos necessários (sub-rede, rede virtual, pool de capacidade) associados. Se o modo de depuração estiver habilitado, Trident registra os recursos do Azure descobertos quando o backend é criado. Verifique se uma função apropriada está sendo usada.

Os valores para `resourceGroups`, `netappAccounts`, `capacityPools`, `virtualNetwork` e `subnet` podem ser especificados usando nomes curtos ou totalmente qualificados. Nomes totalmente qualificados são recomendados na maioria das situações, pois nomes curtos podem corresponder a vários recursos com o mesmo nome.

Se a rede virtual (vNet) estiver localizada em um grupo de recursos diferente da conta de armazenamento do Azure NetApp Files (ANF), especifique o grupo de recursos da rede virtual ao configurar a lista resourceGroups para o backend.

Os `resourceGroups`, `netappAccounts` e `capacityPools` valores são filtros que restringem o conjunto de recursos descobertos àqueles disponíveis para este backend de armazenamento e podem ser especificados em qualquer combinação. Os nomes totalmente qualificados seguem este formato:

Tipo	Formato
Grupo de recursos	<resource group>
Conta NetApp	<resource group>/ cliente NetApp account>

Tipo	Formato
Pool de capacidade	<resource group>/ cliente NetApp account>/<capacity pool>
Rede virtual	<resource group>/<virtual network>
Sub-rede	<resource group>/<virtual network>/<subnet>

Provisionamento de volume

Você pode controlar o provisionamento de volumes padrão especificando as seguintes opções em uma seção especial do arquivo de configuração. Consulte Exemplos de configurações para obter detalhes.

Parâmetro	Descrição	Padrão
`exportRule`	Regras de exportação para novos volumes. `exportRule` Deve ser uma lista separada por vírgulas de qualquer combinação de endereços IPv4 ou sub-redes IPv4 na notação CIDR. Ignorado para volumes SMB.	"0,0.0,0/0"
`snapshotDir`	Acesso ao `.snapshot` diretório	`true`, `false` (Definido explicitamente).
`size`	O tamanho padrão dos novos volumes	"100G"
`unixPermissions`	As permissões unix de novos volumes (4 dígitos octal). Ignorado para volumes SMB.	"" (recurso de pré-visualização, requer lista branca na assinatura)

Exemplos de configurações

Os exemplos a seguir mostram configurações básicas que deixam a maioria dos parâmetros com os valores padrão. Esta é a maneira mais fácil de definir um backend.

▼ **Configuração mínima**

Esta é a configuração mínima absoluta de backend. Com esta configuração, Trident descobre todas as suas NetApp contas, pools de capacidade e sub-redes delegadas ao Azure NetApp Files no local configurado e coloca novos volumes em um desses pools e sub-redes aleatoriamente. Como `nasType` foi omitido, o `nfs` padrão se aplica e o backend irá provisionar volumes NFS.

Essa configuração é ideal quando você está apenas começando a usar o Azure NetApp Files e experimentando as coisas, mas na prática você vai querer fornecer um escopo adicional para os volumes provisionados.

```
---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-anf-1
```

```
  namespace: trident
spec:
  version: 1
  storageDriverName: azure-netapp-files
  subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
  tenantID: 68e4f836-edc1-fake-bff9-b2d865ee56cf
  clientID: dd043f63-bf8e-fake-8076-8de91e5713aa
  clientSecret: SECRET
  location: eastus
```

▼ **Identidades gerenciadas para AKS**

Esta configuração de back-end omits , `subscriptionID` `tenantID`, `clientID`, e `clientSecret`, que são opcionais ao usar identidades gerenciadas.

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-anf-1
  namespace: trident
spec:
  version: 1
  storageDriverName: azure-netapp-files
  capacityPools:
    - resource-group-1/netapp-account-1/ultra-pool
  resourceGroups:
    - resource-group-1
  netappAccounts:
    - resource-group-1/netapp-account-1
  virtualNetwork: resource-group-1/eastus-prod-vnet
  subnet: resource-group-1/eastus-prod-vnet/eastus-anf-subnet
```

▼ **Identidade de nuvem para AKS**

Essa configuração de back-end omits , `tenantID` `clientID`, e `clientSecret`, que são opcionais ao usar uma identidade de nuvem.

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-anf-1
  namespace: trident
spec:
  version: 1
  storageDriverName: azure-netapp-files
  capacityPools:
```

```
    - ultra-pool
  resourceGroups:
    - aks-ami-eastus-rg
  netappAccounts:
    - smb-na
  virtualNetwork: eastus-prod-vnet
  subnet: eastus-anf-subnet
  location: eastus
  subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
```

▼ **Configuração específica de nível de serviço com filtros de pool de capacidade**

Essa configuração de backend coloca volumes na localização do Azure `eastus` em um `Ultra` pool de capacidade. Trident descobre automaticamente todas as sub-redes delegadas ao Azure NetApp Files nesse local e coloca um novo volume em uma delas aleatoriamente.

```
---
version: 1
storageDriverName: azure-netapp-files
subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
tenantID: 68e4f836-edc1-fake-bff9-b2d865ee56cf
clientID: dd043f63-bf8e-fake-8076-8de91e5713aa
clientSecret: SECRET
location: eastus
serviceLevel: Ultra
capacityPools:
  - application-group-1/account-1/ultra-1
  - application-group-1/account-1/ultra-2
```

▼ **Exemplo de backend com pools de capacidade de QoS manuais**

Essa configuração de backend coloca volumes no Azure. `eastus` Localização com pools de capacidade QoS manuais.

```
---
version: 1
storageDriverName: azure-netapp-files
backendName: anf1
location: eastus
labels:
  clusterName: test-cluster-1
  cloud: anf
  nasType: nfs
defaults:
  qosType: Manual
storage:
  - serviceLevel: Ultra
```

```
    labels:
      performance: gold
    defaults:
      maxThroughput: 10
  - serviceLevel: Premium
    labels:
      performance: silver
    defaults:
      maxThroughput: 5
  - serviceLevel: Standard
    labels:
      performance: bronze
    defaults:
      maxThroughput: 3
```

▼ **Configuração avançada**

Essa configuração de back-end reduz ainda mais o escopo do posicionamento de volume para uma única sub-rede e também modifica alguns padrões de provisionamento de volume.

```
---
version: 1
storageDriverName: azure-netapp-files
subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
tenantID: 68e4f836-edc1-fake-bff9-b2d865ee56cf
clientID: dd043f63-bf8e-fake-8076-8de91e5713aa
clientSecret: SECRET
location: eastus
serviceLevel: Ultra
capacityPools:
  - application-group-1/account-1/ultra-1
  - application-group-1/account-1/ultra-2
virtualNetwork: application-group-1/eastus-prod-vnet
subnet: application-group-1/eastus-prod-vnet/my-subnet
networkFeatures: Standard
nfsMountOptions: vers=3,proto=tcp,timeo=600
limitVolumeSize: 500Gi
defaults:
  exportRule: 10.0.0.0/24,10.0.1.0/24,10.0.2.100
  snapshotDir: "true"
  size: 200Gi
  unixPermissions: "0777"
```

▼ **Configuração do pool virtual**

Esta configuração de backend define vários pools de armazenamento em um único arquivo. Isso é útil quando você tem vários pools de capacidade que suportam diferentes níveis de serviço e deseja criar

classes de armazenamento no Kubernetes que os representem. Rótulos de pool virtual foram usados para diferenciar os pools com base em `performance`.

```
---
version: 1
storageDriverName: azure-netapp-files
subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
tenantID: 68e4f836-edc1-fake-bff9-b2d865ee56cf
clientID: dd043f63-bf8e-fake-8076-8de91e5713aa
clientSecret: SECRET
location: eastus
resourceGroups:
  - application-group-1
networkFeatures: Basic
nfsMountOptions: vers=3,proto=tcp,timeo=600
labels:
  cloud: azure
storage:
  - labels:
      performance: gold
    serviceLevel: Ultra
    capacityPools:
      - application-group-1/netapp-account-1/ultra-1
      - application-group-1/netapp-account-1/ultra-2
    networkFeatures: Standard
  - labels:
      performance: silver
    serviceLevel: Premium
    capacityPools:
      - application-group-1/netapp-account-1/premium-1
  - labels:
      performance: bronze
    serviceLevel: Standard
    capacityPools:
      - application-group-1/netapp-account-1/standard-1
      - application-group-1/netapp-account-1/standard-2
```

▼ **Configuração de topologias compatíveis**

Trident facilita o provisionamento de volumes para cargas de trabalho com base em regiões e zonas de disponibilidade. O `supportedTopologies` bloco nesta configuração de backend é usado para fornecer uma lista de regiões e zonas por backend. Os valores de região e zona especificados aqui devem corresponder aos valores de região e zona dos rótulos em cada nó de cluster Kubernetes. Essas regiões e zonas representam a lista de valores permitidos que podem ser fornecidos em uma classe de armazenamento. Para classes de armazenamento que contêm um subconjunto das regiões e zonas fornecidas em um backend, Trident cria volumes na região e zona mencionadas. Para obter mais informações, consulte "Use a topologia CSI".

```
---
version: 1
storageDriverName: azure-netapp-files
subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
tenantID: 68e4f836-edc1-fake-bff9-b2d865ee56cf
clientID: dd043f63-bf8e-fake-8076-8de91e5713aa
clientSecret: SECRET
location: eastus
serviceLevel: Ultra
capacityPools:
  - application-group-1/account-1/ultra-1
  - application-group-1/account-1/ultra-2
supportedTopologies:
  - topology.kubernetes.io/region: eastus
    topology.kubernetes.io/zone: eastus-1
  - topology.kubernetes.io/region: eastus
    topology.kubernetes.io/zone: eastus-2
```

Definições de classe de armazenamento

As definições a seguir `StorageClass` referem-se aos pools de armazenamento acima.

Exemplos de definições usando `parameter.selector` campo

Usando `parameter.selector` você pode especificar para cada `StorageClass` o pool virtual que é usado para hospedar um volume. O volume terá os aspectos definidos no pool escolhido.

```
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gold
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=gold
allowVolumeExpansion: true

---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: silver
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=silver
```

```
allowVolumeExpansion: true

---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: bronze
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=bronze
allowVolumeExpansion: true
```

Definições de exemplo para volumes SMB

Usando `nasType`, `node-stage-secret-name` e `node-stage-secret-namespace`, você pode especificar um volume SMB e fornecer as credenciais necessárias do Active Directory.

▼ **Configuração básica no namespace padrão**

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: anf-sc-smb
provisioner: csi.trident.netapp.io
parameters:
  backendType: "azure-netapp-files"
  trident.netapp.io/nasType: "smb"
  csi.storage.k8s.io/node-stage-secret-name: "smbcreds"
  csi.storage.k8s.io/node-stage-secret-namespace: "default"
```

▼ **Usando diferentes segredos por namespace**

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: anf-sc-smb
provisioner: csi.trident.netapp.io
parameters:
  backendType: "azure-netapp-files"
  trident.netapp.io/nasType: "smb"
  csi.storage.k8s.io/node-stage-secret-name: "smbcreds"
  csi.storage.k8s.io/node-stage-secret-namespace: ${pvc.namespace}
```

▼ **Usando diferentes segredos por volume**

```
apiVersion: storage.k8s.io/v1
```

```
kind: StorageClass
metadata:
  name: anf-sc-smb
provisioner: csi.trident.netapp.io
parameters:
  backendType: "azure-netapp-files"
  trident.netapp.io/nasType: "smb"
  csi.storage.k8s.io/node-stage-secret-name: ${pvc.name}
  csi.storage.k8s.io/node-stage-secret-namespace: ${pvc.namespace}
```

`nasType: smb`Filtros para pools que suportam volumes SMB. `nasType: nfs` ou `nasType: null`filtros para pools NFS.

Crie o backend

Depois de criar o arquivo de configuração de back-end, execute o seguinte comando:

```
tridentctl create backend -f <backend-file>
```

Se você usa uma nuvem Azure não comercial, certifique-se de que `tridentctl` está configurado para usar o Azure Resource Manager e os endpoints de autenticação para o seu ambiente de nuvem Azure. Se a criação do backend falhar, verifique a configuração do backend e visualize os logs para determinar a causa:

```
tridentctl logs
```

Depois de identificar e corrigir o problema com o arquivo de configuração, você pode executar o comando create novamente.

Google Cloud NetApp volumes

Configurar Google Cloud NetApp Volumes

Você pode configurar o Google Cloud NetApp Volumes como um backend para Trident a fim de provisionar storage para cargas de trabalho do Kubernetes.

Visão geral

Trident oferece suporte ao Google Cloud NetApp Volumes para cargas de trabalho NAS (NFS e SMB) e em bloco (iSCSI).

- As cargas de trabalho NAS usam o `google-cloud-netapp-volumes` backend
- As cargas de trabalho em bloco (iSCSI) usam o `google-cloud-netapp-volumes-san`backend

Os volumes NAS fornecem armazenamento baseado em arquivos e são acessados usando os protocolos NFS ou SMB. Esses volumes suportam acesso compartilhado entre vários pods ou nós.

Os volumes de bloco fornecem armazenamento bruto em bloco e são acessados como dispositivos iSCSI

conectados a nós do Kubernetes. Esses volumes são usados quando os aplicativos exigem acesso em nível de bloco.

Isso se aplica aos seguintes ambientes:

- Trident 26.02 e posteriores
- Google Kubernetes Engine (GKE) ou clusters do Red Hat OpenShift em execução no GCP
- OpenShift A Federação de Identidade de Carga de Trabalho (WIF) requer o Trident 26.10 ou posterior
- Pools de armazenamento do Google Cloud NetApp Volumes

Para configurar o armazenamento em bloco (iSCSI), consulte "Configurar armazenamento em bloco (iSCSI)".

Prepare-se para configurar

A identidade na nuvem permite que as cargas de trabalho do Kubernetes acessem recursos do Google Cloud autenticando-se como uma identidade de carga de trabalho em vez de usar credenciais estáticas.

Antes de configurar a identidade da carga de trabalho, use uma conta autorizada a criar os recursos de identidade necessários. Para o GKE, isso requer permissões do Google Cloud para criar e configurar uma conta de serviço do Google Cloud. Para o OpenShift, isso requer permissão para criar ou selecionar a configuração de WIF do cluster e para configurar o Google Cloud Workload Identity Pool e o provedor.

Usar a identidade de carga de trabalho do GKE

Para usar a identidade de workload do GKE com Google Cloud NetApp Volumes, você precisa ter:

- Um cluster Kubernetes implantado usando Google Kubernetes Engine (GKE)
- Identidade de carga de trabalho habilitada no cluster GKE e o servidor de metadados habilitado nos pools de nós
- Uma conta de serviço do Google Cloud com a função de Administrador do Google Cloud NetApp Volumes (`roles/netapp.admin`) ou uma função personalizada equivalente
- Trident instalado com o provedor de nuvem definido como `GCP` e a anotação de identidade da nuvem configurada

Operador Trident

Para instalar o Trident usando o operador Trident, edite `tridentorchestrator_cr.yaml`:

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  namespace: trident
  cloudProvider: "GCP"
  cloudIdentity: "iam.gke.io/gcp-service-account: cloudvolumes-admin-
sa@mygcpproject.iam.gserviceaccount.com"
```

Leme

Defina o provedor de nuvem e a identidade da nuvem ao instalar o Trident com o Helm:

```
helm install trident trident-operator-100.6.0.tgz \
  --set cloudProvider=GCP \
  --set cloudIdentity="iam.gke.io/gcp-service-account: cloudvolumes-
admin-sa@mygcpproject.iam.gserviceaccount.com"
```

tridentctl

Instale Trident especificando o provedor de nuvem e a identidade de nuvem:

```
tridentctl install \
  --cloud-provider=GCP \
  --cloud-identity="iam.gke.io/gcp-service-account: cloudvolumes-admin-
sa@mygcpproject.iam.gserviceaccount.com" \
  -n trident
```

Use a Federação de Identidade da Carga de Trabalho no OpenShift

A partir do Trident 26.10, você pode usar o Google Cloud Workload Identity Federation (WIF) para autenticar os drivers do Google Cloud NetApp Volumes em clusters do OpenShift executados no GCP. Isso se aplica ao OpenShift Container Platform (OCP, incluindo implantações IPI) autogerenciado e ao OpenShift Dedicated (OSD), e oferece suporte aos drivers NAS (`google-cloud-netapp-volumes`) e SAN (`google-cloud-netapp-volumes-san`).

O WIF troca um token projetado de conta de serviço do Kubernetes por credenciais de curta duração do Google Cloud. O Trident usa essas credenciais para representar uma conta de serviço do Google Cloud; as funções do IAM dessa conta de serviço autorizam suas chamadas de API do Google Cloud NetApp Volumes. Isso evita armazenar uma chave de conta de serviço de longa duração na configuração de backend.

> No OpenShift Dedicated, o provedor OIDC pode ser compartilhado com outros componentes do OpenShift. Se a lista `allowedAudiences` do provedor estiver vazia, o Google aceita o nome canônico do recurso do provedor, com ou sem o `https://`prefixo, portanto nenhuma alteração no provedor é necessária para o público do Trident. Não defina uma lista explícita de público apenas para o Trident. Se o provedor já tiver públicos explícitos, acrescente o público do Trident, preservando todos os valores existentes. Substituir a lista pode impedir que outros componentes do OpenShift troquem tokens.

Prepare o cluster do OpenShift

Para OSD, crie ou selecione uma configuração WIF do GCP ao criar o cluster. Por exemplo, a OpenShift Cluster Manager CLI pode criar uma:

```
ocm gcp create wif-config --name <wif-name> --project <gcp-project-id>
```

Selecione Workload Identity Federation como o tipo de autenticação do cluster e associe a configuração WIF

ao cluster. Para o OCP autogerenciado, configure um provedor OIDC do Google Cloud que confie no emissor do token da conta de serviço do cluster. As etapas de provisionamento do cluster e de projeção do token diferem entre o OSD e o OCP autogerenciado; siga as orientações aplicáveis da Red Hat e disponibilize o token projetado para o controlador do Trident em `/var/run/secrets/wif-token/token`.

A autenticação WIF do OSD é configurada quando você cria o cluster. O público WIF do Trident é uma configuração de instalação separada; você não precisa recriar o cluster para habilitar o WIF para o Trident.

Antes de instalar ou configurar o Trident, verifique se a identidade do OpenShift usada para a instalação tem as permissões de cluster exigidas pelo método de instalação que você escolheu. Após a instalação, confirme se os pods do controlador e dos nós estão saudáveis:

```
oc whoami
oc auth can-i create namespaces
oc auth can-i create csidrivers.storage.k8s.io
oc auth can-i create customresourcedefinitions.apiextensions.k8s.io
oc auth can-i create securitycontextconstraints.security.openshift.io

oc -n trident get pods
oc -n trident rollout status deploy/trident-controller --timeout=300s
oc -n trident rollout status ds/trident-node-linux --timeout=300s
```

Instale o Trident com o público do WIF

O Trident projeta o token para WIF somente quando você configura o público-alvo durante a instalação. Use o mesmo público-alvo no provedor do Google Cloud e no do backend `wipCredentialConfig.audience`:

```
//iam.googleapis.com/projects/<wif-pool-project-
number>/locations/global/workloadIdentityPools/<pool-
id>/providers/<provider-id>
```

Defina `cloudProvider` como `GCP` e configure `gcpWIFAudience` usando o método de instalação que você selecionou:

tridentctl

```
tridentctl install \
  --cloud-provider GCP \
  --gcp-wif-audience "<audience>" \
  --namespace trident
```

Operador Trident

Defina esses campos na `TridentOrchestrator` especificação:

```
spec:
```

```
  cloudProvider: GCP
  gcpWIFAudience: "<audience>"
```

Leme

Defina estes valores ao instalar o chart:

```
helm install trident netapp-trident/trident-operator \
  --set cloudProvider=GCP \
  --set-string gcpWIFAudience="<audience>"
```

Não configure também a opção GKE `cloudIdentity` / `--cloud-identity` para o WIF do OpenShift. Não aplique patch no Deployment do controlador do Trident após a instalação; o operador reverte as alterações manuais. O token projetado está disponível para o backend em `/var/run/secrets/wif-token/token`. O `trident-main` contêiner não inclui um shell, portanto inspecione a configuração de volume do Deployment do controlador em vez de usar `oc exec` para verificar o arquivo:

```
oc -n trident get deploy trident-controller -o yaml
```

Configure a confiança e as permissões do Google Cloud

Configure um Google Cloud Workload Identity Pool e um provedor OIDC para o cluster. O emissor do provedor deve corresponder ao claim `iss` do token, e o mapeamento de atributos deve mapear `assertion.sub` para `google.subject`. O público permitido do provedor deve corresponder ao claim `aud` do token projetado.

Os endpoints públicos `.well-known/openid-configuration` e JWKS são necessários apenas ao usar a descoberta OIDC. O Google também oferece suporte ao upload direto do JWKS usando `jwksJson` ou `--jwk-json-path`.

Atribua à entidade principal WIF a `roles/iam.workloadIdentityUser`função na conta de serviço do Google Cloud que o Trident vai representar. Para um sujeito de token de `system:serviceaccount:trident:trident-controller`, a entidade principal tem este formato:

```
principal://iam.googleapis.com/projects/<wif-pool-project-
number>/locations/global/workloadIdentityPools/<pool-
id>/subject/system:serviceaccount:trident:trident-controller
```

Conceda à conta de serviço representada `roles/netapp.admin` no projeto que contém os recursos do Google Cloud NetApp Volumes, ou uma função personalizada equivalente com as permissões necessárias. Essas são concessões separadas: a primeira permite a representação, e a segunda autoriza as operações da API de storage.

Se a configuração do público-alvo e do provedor do token não corresponder, o Serviço de Token de Segurança do Google poderá rejeitar a troca com um `invalid_grant` erro de público-alvo. Verifique a configuração do provedor com:

```
gcloud iam workload-identity-pools providers describe <provider-id> \
  --project=<wif-pool-project-id> \
  --location=global \
  --workload-identity-pool=<pool-id> \
  --format="yaml(oidc.allowedAudiences,oidc.issuerUri,attributeMapping)"
```

Ao diagnosticar a configuração do token, compare as declarações de `iss`, `aud` e `sub` do token com a configuração do provedor e do IAM. Não compartilhe nem registre o token projetado completo.

Prepare o armazenamento do Google Cloud NetApp Volumes

Crie ou identifique um pool de storage na região usada pelo backend. Os backends NAS exigem um pool FILE; os backends SAN usam os pools Flex Unified descritos em "Configurar armazenamento em bloco (iSCSI)". Certifique-se de que a configuração de VPC e Private Service Access (PSA) do pool seja adequada para a rede do cluster.

Por exemplo, liste os intervalos de peering VPC globais existentes para uma rede antes de selecionar o intervalo PSA para um pool:

```
gcloud compute addresses list \
  --project <gcnv-project-id> \
  --global \
  --filter="purpose=VPC_PEERING AND network:<network-name>" \
  --format="table(name,address,prefixLength,status,network,purpose)"
```

Use a região do pool em `location` e, quando necessário, selecione o pool explicitamente usando seu caminho de recurso completo em `storagePools`. Incompatibilidades entre a região e a localização do pool, uma rede incorreta ou um pool indisponível podem impedir a descoberta ou o provisionamento, mesmo quando a autenticação é bem-sucedida.

Configure o backend do Trident

Use o público-alvo projetado do token tanto na configuração do provedor quanto em `wipCredentialConfig.audience`. O `projectNumber` campo identifica o projeto do Google Cloud NetApp Volumes; o público-alvo usa o número do projeto que contém o Workload Identity Pool. Substitua os espaços reservados neste exemplo pelos valores do seu ambiente:

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: gcnv-nas-wif
  namespace: trident
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: "<project-number>"
  location: "<region>"
```

```
  sdkTimeout: "600"
  storage:
  - labels:
      cloud: gcp
    network: "<vpc-network>"
    serviceLevel: Premium
  wipCredentialConfig:
    type: external_account
    audience: "//iam.googleapis.com/projects/<wif-pool-project-
number>/locations/global/workloadIdentityPools/<pool-
id>/providers/<provider-id>"
    subject_token_type: urn:ietf:params:oauth:token-type:jwt
    token_url: https://sts.googleapis.com/v1/token
    credential_source:
      file: /var/run/secrets/wif-token/token
    service_account_impersonation_url:
https://iamcredentials.googleapis.com/v1/projects/-
/serviceAccounts/<service-account-email>:generateAccessToken
```

Para um backend SAN, use `google-cloud-netapp-volumes-san` como `storageDriverName` e as configurações aplicáveis do pool Flex Unified. Fora isso, a configuração da credencial WIF é a mesma para ambos os drivers.

Aplique o backend e verifique o status observado:

```
oc apply -f <backend-file>.yaml
oc -n trident get tridentbackendconfig <backend-name> -o yaml
```

Confirme se o backend está `Bound` e se a última operação foi bem-sucedida antes de testar o provisionamento. Inspecione o ambiente ativo `spec` e `status` a configuração; uma anotação de configuração aplicada anteriormente pode não refletir alterações posteriores. Não inclua credenciais estáticas `apiKey` ao usar o WIF.

Verifique o provisionamento completamente

Crie uma StorageClass que tenha como destino o backend e, em seguida, um pequeno PVC de teste. Por exemplo, uma StorageClass NAS pode selecionar um rótulo de backend:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gcnv-wif-smoke
provisioner: csi.trident.netapp.io
parameters:
  backendType: "google-cloud-netapp-volumes"
  selector: "auth=wif"
```

```
allowVolumeExpansion: true
```

Adicione o rótulo correspondente `auth: wif` à entrada de storage do backend NAS, crie um PVC usando `gcnv-wif-smoke` e verifique se ele alcança `Bound`. Para um teste rápido de SAN, use `google-cloud-netapp-volumes-san`, as configurações do driver SAN e um pool Flex Unified apropriado. Exclua o PVC de teste e a StorageClass ao concluir.

Solucione falhas comuns

Sintoma	O que verificar
`invalid_grant` ou incompatibilidade de público	Compare o token projetado `aud` com os públicos permitidos do provedor e `wipCredentialConfig.audience`.
`iam.serviceAccounts.getAccessToken` permissão negada	Verifique se a principal WIF exata do controlador do Trident `sub` tem `roles/iam.workloadIdentityUser` na conta de serviço representada.
`netapp.storagePools.list` permissão negada	Verifique se a conta de serviço representada possui as permissões do Google Cloud NetApp Volumes, como `roles/netapp.admin`, no projeto correto.
Nenhum pool de storage encontrado ou pool de storage não encontrado	Verifique o backend `location`, a região e o tipo do pool, a rede, a configuração do PSA e qualquer caminho de recurso do pool de storage configurado explicitamente.
O backend permanece desvinculado	Inspecione os logs do controlador do Trident e o status de TridentBackendConfig para identificar o erro subjacente de autenticação, permissão ou descoberta de pool.

Para inspecionar os logs do controlador em busca de erros de reconciliação no backend:

```
oc -n trident logs deploy/trident-controller -c trident-main --tail=400
```

Configurar armazenamento NAS

> Para pools de armazenamento UNIFIED do Google Cloud NetApp Volumes, Trident aplica regras de nomenclatura e validação específicas do UNIFIED durante as operações de volume.

Ao localizar um volume, Trident pode avaliar várias variantes de nomes de volume compatíveis (por exemplo, formatos com hífen e sublinhado) para melhorar a confiabilidade da importação e da descoberta.

Detalhes do driver

Trident fornece o `google-cloud-netapp-volumes` driver para provisionar armazenamento NAS a partir do Google Cloud NetApp Volumes.

O driver suporta os seguintes modos de acesso:

- ReadWriteOnce (RWO)
- ReadOnlyMany (ROX)

- ReadWriteMany (RWX)
- ReadWriteOncePod (RWOP)

Condutor	Protocolo	VolumeMode	Modos de acesso suportados	Sistemas de arquivos suportados
`google-cloud-netapp-volumes`	NFS, SMB	Sistema de ficheiros	RWO, ROX, RWX, RWOP	`nfs`, `smb`

Configurar um backend NAS do Trident

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: gcnv-nas
  namespace: trident
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: "<project-number>"
  location: "<region>"
  sdkTimeout: "600"
  storage:
  - labels:
      cloud: gcp
    network: "<vpc-network>"
```

Provisionar volumes NAS

Os volumes NAS são provisionados usando o `google-cloud-netapp-volumes` backend e suportam os protocolos NFS e SMB.

StorageClass para volumes NFS

Para provisionar volumes NFS, defina `nasType` para `nfs`.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gcnv-nfs
provisioner: csi.trident.netapp.io
parameters:
  backendType: "google-cloud-netapp-volumes"
  trident.netapp.io/nasType: "nfs"
allowVolumeExpansion: true
```

StorageClass for volumes SMB

Para provisionar volumes SMB, defina `nasType` para `smb` e forneça as credenciais.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gcnv-smb
provisioner: csi.trident.netapp.io
parameters:
  backendType: "google-cloud-netapp-volumes"
  trident.netapp.io/nasType: "smb"
  csi.storage.k8s.io/node-stage-secret-name: "smbcreds"
  csi.storage.k8s.io/node-stage-secret-namespace: "default"
allowVolumeExpansion: true
```

Exemplo de PersistentVolumeClaim (RWX)

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: gcnv-nas-rwx
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 100Gi
  storageClassName: gcnv-nfs
```

Exemplo de PersistentVolumeClaim (RWO)

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: gcnv-nas-rwo
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 100Gi
  storageClassName: gcnv-nfs
```

Volumes NAS usam `volumeMode: Filesystem`.

Configurar o Google Cloud NetApp Volumes para cargas de trabalho SAN

Use o `google-cloud-netapp-volumes-san` driver para provisionar volumes de bloco iSCSI dos pools de armazenamento Flex Unified do Google Cloud NetApp Volumes.

O driver SAN suporta apenas cargas de trabalho em bloco. Para provisionar volumes NAS, use o `google-cloud-netapp-volumes` driver.

Visão geral

O driver provisiona volumes dos pools de armazenamento do Flex Unified e os apresenta aos nós do Kubernetes como dispositivos de bloco iSCSI.

O driver é compatível com os seguintes ambientes:

- Trident 26.02 e posteriores
- Google Kubernetes Engine (GKE) ou clusters do Red Hat OpenShift em execução no GCP
- Pools de storage unificado do Google Cloud NetApp Volumes Flex
- Cargas de trabalho baseadas em iSCSI

A partir do o Trident 26.10, o driver SAN oferece suporte ao Google Cloud Workload Identity Federation (WIF) na plataforma autogerenciada OpenShift Container Platform (OCP, incluindo implantações IPI) e em clusters OpenShift Dedicated (OSD) executados no GCP. Para conhecer os pré-requisitos do WIF e a configuração do backend, consulte "Use a Federação de Identidade da Carga de Trabalho no OpenShift". Use o nome do driver SAN e as configurações de armazenamento do o Flex Unified nesta página.

Configure a WIF para SAN

O WIF troca um token de conta de serviço projetado do Kubernetes por credenciais de curta duração do Google Cloud. O Trident usa essas credenciais para se passar por uma conta de serviço do Google Cloud ao chamar a API do Google Cloud NetApp Volumes.

Configure um pool de identidades de workload do Google Cloud e um provedor OIDC para o cluster OpenShift. O emissor do provedor deve corresponder ao claim `iss` do token, e o mapeamento de atributos deve mapear `assertion.sub` para `google.subject`.

Instale o Trident com o nome do recurso do provedor como o público do WIF. Isso fará com que o Trident projete o token da conta de serviço do controlador em `/var/run/secrets/wif-token/token`.

Operador Trident

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  namespace: trident
```

```
  cloudProvider: GCP
  gcpWIFAudience: "//iam.googleapis.com/projects/<wif-pool-project-
number>/locations/global/workloadIdentityPools/<pool-
id>/providers/<provider-id>"
```

Leme

```
helm install trident <trident-operator-package> \
  --namespace trident \
  --create-namespace \
  --set cloudProvider=GCP \
  --set-string gcpWIFAudience="//iam.googleapis.com/projects/<wif-pool-
project-number>/locations/global/workloadIdentityPools/<pool-
id>/providers/<provider-id>"
```

tridentctl

```
tridentctl install \
  --namespace trident \
  --cloud-provider GCP \
  --gcp-wif-audience "//iam.googleapis.com/projects/<wif-pool-project-
number>/locations/global/workloadIdentityPools/<pool-
id>/providers/<provider-id>"
```

Use o mesmo público-alvo para a instalação do Trident e `wipCredentialConfig.audience`. Também não configure o GKE `cloudIdentity` ou `--cloud-identity`. Não aplique patches no Deployment do controlador para adicionar o token projetado; o operador do Trident pode sobrescrever essa alteração.

> No OpenShift Dedicated, o provedor OIDC pode ser compartilhado com outros componentes do OpenShift. Se a lista do provedor `allowedAudiences` estiver vazia, o Google aceita o nome canônico do recurso do provedor, com ou sem o `https://` prefixo, portanto nenhuma alteração no provedor é necessária para o público acima. Não defina uma lista de público explícita apenas para o Trident. Se o provedor já tiver públicos explícitos, acrescente o público do Trident, preservando todos os valores existentes. Substituir a lista pode impedir que outros componentes do OpenShift troquem tokens.

Conceda à entidade principal do WIF da conta de serviço do controlador do Trident a função `roles/iam.workloadIdentityUser` na conta de serviço do Google Cloud que o Trident representará. Para a conta de serviço padrão do controlador do Trident, o assunto do token é `system:serviceaccount:trident:trident-controller`; use o assunto configurado para sua instalação. Conceda à conta de serviço representada a função `roles/netapp.admin`, ou permissões equivalentes, no projeto que contém os recursos do Google Cloud NetApp Volumes.

Adicione o seguinte bloco ao `spec` do backend SAN:

```
wipCredentialConfig:
  type: external_account
```

```
  audience: "//iam.googleapis.com/projects/<wif-pool-project-
number>/locations/global/workloadIdentityPools/<pool-
id>/providers/<provider-id>"
  subject_token_type: urn:ietf:params:oauth:token-type:jwt
  token_url: https://sts.googleapis.com/v1/token
  credential_source:
    file: /var/run/secrets/wif-token/token
  service_account_impersonation_url:
https://iamcredentials.googleapis.com/v1/projects/-
/serviceAccounts/<service-account-email>:generateAccessToken
```

A configuração WIF é a mesma para os dois drivers GCNV. O backend SAN ainda deve usar `google-cloud-netapp-volumes-san` e um pool de storage Flex Unified compatível.

Pools de storage unificado Flex

Os pools de armazenamento Flex Unified fornecem armazenamento em bloco usando o protocolo iSCSI e são necessários para o provisionamento de SAN:

- Os pools regionais Flex Unified são suportados.
- Os pools Zonais unificados Flex são suportados a partir do Trident 26.02.1.
- Apenas o nível de serviço **Flex** é compatível com cargas de trabalho SAN.

O backend `location` deve corresponder à região do pool Flex Unified. Se o Trident não encontrar o pool, verifique a região, a rede e o caminho do recurso do pool.

Configurar um backend Trident SAN

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: gcnv-san
  namespace: trident
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes-san
  projectNumber: "<project-number>"
  location: "<region>"
  sdkTimeout: "600"
  storage:
  - labels:
      cloud: gcp
      performance: flex
    network: "<vpc-network>"
    serviceLevel: Flex
```

Crie um StorageClass

Após configurar o backend SAN, crie um StorageClass que faça referência ao `google-cloud-netapp-volumes-san` driver.

Especifique o tipo de sistema de arquivos no StorageClass, não no backend.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gcnv-san
provisioner: csi.trident.netapp.io
parameters:
  backendType: "google-cloud-netapp-volumes-san"
  fsType: "ext4"
allowVolumeExpansion: true
```

O driver é compatível com os seguintes tipos de sistema de arquivos:

- `ext4` (padrão)
- `ext3`
- `xfs`

> O driver SAN suporta apenas o nível de serviço Flex e não utiliza parâmetros de backend específicos do NAS, como `exportRule`, `unixPermissions`, `nasType`, `snapshotDir`, `nfsMountOptions` ou configurações relacionadas ao tiering.

Verificar o provisionamento de WIF

Aplique o backend e verifique se o TridentBackendConfig está vinculado e se sua última operação foi bem-sucedida:

```
oc apply -f <backend-file>.yaml
oc -n trident get tridentbackendconfig <backend-name> -o yaml
```

Crie um PVC usando a `gcnv-san` StorageClass desta página e verifique se ele chega a `Bound`. Isso valida em conjunto a troca de WIF, a representação da conta de serviço, as permissões do Google Cloud NetApp Volumes e a seleção do pool SAN. Exclua o PVC de teste quando terminar.

Se a reconciliação do backend falhar, inspecione os logs do controlador do Trident:

```
oc -n trident logs deploy/trident-controller -c trident-main --tail=400
```

Verifique o seguinte quanto a erros comuns:

- Para um `invalid_grant` erro de público-alvo, compare a declaração `aud` do token com os públicos-alvo

permitidos pelo provedor OIDC e `wipCredentialConfig.audience`.

- Para um `iam.serviceAccounts.getAccessToken` erro de permissão, verifique se o principal WIF do controlador do Trident tem `roles/iam.workloadIdentityUser` na conta de serviço representada.
- Em caso de `netapp.storagePools.list` erro de permissão, verifique se a conta de serviço representada tem as permissões necessárias do Google Cloud NetApp Volumes no projeto correto.
- Caso o pool não seja encontrado, verifique se sua localização e tipo são compatíveis, se a região do backend corresponde à região do pool e se o backend está referenciando a rede e o pool corretos.

Provisionar volumes de bloco

ReadWriteOnce (RWO)

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: gcnv-san-rwo
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 100Gi
  storageClassName: gcnv-san
```

ReadWriteOncePod (RWOP)

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: gcnv-san-rwop
spec:
  accessModes:
    - ReadWriteOncePod
  resources:
    requests:
      storage: 100Gi
  storageClassName: gcnv-san
```

ReadOnlyMany (ROX)

Um padrão comum para o ROX é clonar um volume ReadWriteOnce existente e montar o clone como somente leitura.

```
apiVersion: v1
```

```
kind: PersistentVolumeClaim
metadata:
  name: gcnv-san-rox
spec:
  accessModes:
    - ReadOnlyMany
  resources:
    requests:
      storage: 100Gi
  storageClassName: gcnv-san
  dataSource:
    kind: PersistentVolumeClaim
    name: gcnv-san-rwo
```

ReadWriteMany (RWX) — somente bloco bruto

ReadWriteMany é suportado somente quando `volumeMode: Block`.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: gcnv-san-raw-rwx
spec:
  accessModes:
    - ReadWriteMany
  volumeMode: Block
  resources:
    requests:
      storage: 100Gi
  storageClassName: gcnv-san
```

Comportamento do volume do bloco

Os volumes de bloco são provisionados como LUNs iSCSI e apresentados aos nós do Kubernetes como dispositivos de bloco.

Volumes de bloco:

- Use o protocolo iSCSI
- Suporte ao sistema de arquivos e apresentação de blocos brutos
- Estão anexados e gerenciados pela Trident
- Suporte a múltiplos modos de acesso do Kubernetes

Modos de acesso

Os volumes de bloco provisionados pelo Trident suportam os seguintes modos de acesso:

- `ReadWriteOnce` (RWO)
- `ReadOnlyMany` (ROX)
- `ReadWriteOncePod` (RWOP)
- `ReadWriteMany` (RWX), suportado somente quando `volumeMode: Block`

Comportamento do volumeMode

O `volumeMode` campo controla como um volume de bloco é exposto:

- `Filesystem` Trident formata e monta o volume.
- `Block` Trident conecta o dispositivo e o expõe como um dispositivo de bloco raw.

Operações suportadas

Volumes de bloco provisionados usando o driver `google-cloud-netapp-volumes-san` oferecem suporte a:

- Criar
- Excluir
- Clonar
- Snapshot
- Redimensionar
- Importar

Sobrecarga do tamanho do volume

Os volumes em bloco do Google Cloud NetApp Volumes incluem sobrecarga de metadados internos. Como resultado, o tamanho do dispositivo visível para o kernel é menor do que a capacidade provisionada.

Sobrecarga observada:

- Aproximadamente 300 KiB quando um volume é criado
- Até aproximadamente 107 MiB após um redimensionamento

O Google Cloud NetApp Volumes aceita alocações de capacidade em GiB inteiros. Para garantir que o tamanho utilizável do dispositivo atenda ou exceda a solicitação do PVC, o Trident:

- Arredonda o tamanho solicitado para o próximo GiB inteiro.
- Adiciona um buffer de 1 GiB.

Por exemplo:

- Pedido de PVC: 100 GiB.
- Capacidade provisionada: 101 GiB.
- Capacidade utilizável visível para a aplicação: pelo menos 100 GiB.

Exemplos de pods

Volume de bloco montado no sistema de arquivos (RWO)

```
apiVersion: v1
kind: Pod
metadata:
  name: app-rwo
spec:
  containers:
  - name: app
    image: ubuntu:22.04
    command: ["sleep", "infinity"]
    volumeMounts:
    - name: data
      mountPath: /mnt/data
  volumes:
  - name: data
    persistentVolumeClaim:
      claimName: gcnv-san-rwo
```

Dispositivo de bloco bruto (RWX)

```
apiVersion: v1
kind: Pod
metadata:
  name: app-raw-rwx
spec:
  containers:
  - name: app
    image: ubuntu:22.04
    command: ["sleep", "infinity"]
    volumeDevices:
    - name: data
      devicePath: /dev/xda
  volumes:
  - name: data
    persistentVolumeClaim:
      claimName: gcnv-san-raw-rwx
```

Comportamento de attach e montagem

Para volumes SAN provisionados a partir do Google Cloud NetApp Volumes:

- Trident cria um Número de Unidade Lógica (LUN) em um pool de storage Flex Unified.

- Durante a publicação, Trident mapeia o LUN para um grupo de hosts por nó.
- Durante o preparo do nó, Trident:
  - Faz login no destino iSCSI
  - Descobre o LUN
  - Configura multipath
- Se `volumeMode: Filesystem`, o Trident formata o dispositivo, se necessário, e o monta.
- Se `volumeMode: Block`Trident anexar o dispositivo e expô-lo diretamente ao pod sem formatar ou montar.

> Os volumes de bloco SAN não oferecem bloqueio distribuído nem coordenação de gravação. Quando um volume de bloco é acessado por vários nós (ReadWriteMany com `volumeMode: Block`), o aplicativo ou o sistema de arquivos deve gerenciar a concorrência.

Prepare-se para configurar um back-end do Google Cloud NetApp volumes

Antes de configurar o back-end do Google Cloud NetApp volumes, você precisa garantir que os requisitos a seguir sejam atendidos.

Pré-requisitos para volumes NFS ou SMB

Se você estiver usando o Google Cloud NetApp Volumes pela primeira vez ou em um novo local, será necessária alguma configuração inicial para configurar o Google Cloud NetApp Volumes e criar um volume NFS ou SMB. Consulte "Antes de começar".

Antes de configurar o back-end do Google Cloud NetApp volumes:

- Uma conta do Google Cloud configurada com o serviço Google Cloud NetApp volumes. "Google Cloud NetApp volumes"Consulte a .
- Número do projeto da sua conta do Google Cloud. "Identificação de projetos"Consulte a .
- Uma conta de serviço do Google Cloud com a (`roles/netapp.admin`função de administrador do NetApp volumes). "Funções e permissões de gerenciamento de identidade e acesso"Consulte a .
- Arquivo de chave de API para sua conta GCNV. Consulte "Crie uma chave de conta de serviço"
- Um pool de armazenamento. "Visão geral dos pools de armazenamento"Consulte a .

Para obter mais informações sobre como configurar o acesso ao Google Cloud NetApp volumes, "Configurar o acesso ao Google Cloud NetApp volumes"consulte .

Exemplos e opções de configuração de back-end do Google Cloud NetApp volumes

Saiba mais sobre as opções de configuração de back-end para o Google Cloud NetApp volumes e revise exemplos de configuração.

Opções de configuração de back-end

Cada back-end provisiona volumes em uma única região do Google Cloud. Para criar volumes em outras regiões, você pode definir backends adicionais.

Parâmetro	Descrição	Padrão
`version`		Sempre 1
`storageDriverName`	Nome do controlador de armazenamento	O valor de `storageDriverName` deve ser especificado como "google-cloud-NetApp-volumes".
`backendName`	(Opcional) Nome personalizado do back-end de armazenamento	Nome do driver e parte da chave da API
`storagePools`	Parâmetro opcional usado para especificar pools de armazenamento para criação de volume.	
`projectNumber`	Número do projeto da conta Google Cloud. O valor é encontrado na página inicial do portal do Google Cloud.	
`location`	O Trident cria volumes de GCNV. Ao criar clusters de Kubernetes entre regiões, os volumes criados em a `location` podem ser usados em workloads programados em nós em várias regiões do Google Cloud. O tráfego entre regiões incorre em um custo adicional.	
`apiKey`	Chave de API para a conta de serviço do Google Cloud com a `netapp.admin` função. Ele inclui o conteúdo formatado em JSON do arquivo de chave privada de uma conta de serviço do Google Cloud (copiado literalmente no arquivo de configuração de back-end). O `apiKey` deve incluir pares de chave-valor para as seguintes chaves: `type` `project_id` , , `client_email` `client_id` , , , , `auth_uri` `token_uri`, `auth_provider_x509_cert_url`, e `client_x509_cert_url`.	
`nfsMountOptions`	Controle refinado das opções de montagem NFS.	"3"
`limitVolumeSize`	Falha no provisionamento se o tamanho do volume solicitado estiver acima desse valor.	"" (não aplicado por padrão)
`serviceLevel`	O nível de serviço de um pool de storage e seus volumes. Os valores são `flex`, `standard`, `premium`, `extreme` ou .	
`labels`	Conjunto de rótulos arbitrários formatados em JSON para aplicar em volumes	""
`network`	Rede do Google Cloud usada para volumes GCNV.	
`debugTraceFlags`	Debug flags para usar ao solucionar problemas. Exemplo, `{"api":false, "method":true}`. Não use isso a menos que você esteja solucionando problemas e exija um despejo de log detalhado.	nulo

Parâmetro	Descrição	Padrão
`nasType`	Configurar a criação de volumes NFS ou SMB. As opções são `nfs`, `smb` ou null. A configuração como null padrão para volumes NFS.	`nfs`
`supportedTopologies`	Representa uma lista de regiões e zonas que são suportadas por este backend. Para obter mais informações, "Use a topologia CSI"consulte . Por exemplo: `supportedTopologies: - topology.kubernetes.io/region: asia-east1 topology.kubernetes.io/zone: asia-east1-a`	

Opções de provisionamento de volume

Você pode controlar o provisionamento de volume padrão `defaults` na seção do arquivo de configuração.

Parâmetro	Descrição	Padrão
`exportRule`	As regras de exportação para novos volumes. Deve ser uma lista separada por vírgulas de qualquer combinação de endereços IPv4.	"0,0.0,0/0"
`snapshotDir`	Acesso ao `.snapshot` diretório	`true`, `false` (o comportamento padrão pode variar. Defina explicitamente) "false" para NFSv3
`snapshotReserve`	Porcentagem de volume reservado para snapshots	"" (aceitar predefinição de 0)
`unixPermissions`	As permissões unix de novos volumes (4 dígitos octal).	""

Exemplos de configurações

Os exemplos a seguir mostram configurações básicas que deixam a maioria dos parâmetros padrão. Esta é a maneira mais fácil de definir um backend.

▼ **Configuração mínima**

Esta é a configuração mínima absoluta de back-end. Com essa configuração, o Trident descobre todos os pools de armazenamento delegados ao Google Cloud NetApp volumes no local configurado e coloca novos volumes aleatoriamente em um desses pools. Como `nasType` é omitido, o `nfs` padrão se aplica e o back-end provisionará para volumes NFS.

Essa configuração é ideal para quem está começando a usar o Google Cloud NetApp Volumes e testando seus recursos, mas na prática pode ser necessário definir um escopo adicional para os volumes provisionados.

Substitua `<id_value>` e `<key_value>` pelas credenciais da sua conta de serviço.

```
---
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-gcnv-secret
type: Opaque
stringData:
  private_key_id: "<id_value>"
  private_key: |
    -----BEGIN PRIVATE KEY-----
    <key_value>
    -----END PRIVATE KEY-----
---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-gcnv
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: "123455380079"
  location: europe-west6
  serviceLevel: premium
  apiKey:
    type: service_account
    project_id: my-gcnv-project
    client_email: myproject-prod@my-gcnv-project.iam.gserviceaccount.com
    client_id: "103346282737811234567"
    auth_uri: https://accounts.google.com/o/oauth2/auth
    token_uri: https://oauth2.googleapis.com/token
    auth_provider_x509_cert_url:
https://www.googleapis.com/oauth2/v1/certs
    client_x509_cert_url:
https://www.googleapis.com/robot/v1/metadata/x509/myproject-prod%40my-
gcnv-project.iam.gserviceaccount.com
  credentials:
    name: backend-tbc-gcnv-secret
```

▼ Configuração para volumes SMB

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-gcnv1
  namespace: trident
```

```
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: "123456789"
  location: asia-east1
  serviceLevel: flex
  nasType: smb
  apiKey:
    type: service_account
    project_id: cloud-native-data
    client_email: trident-sample@cloud-native-
data.iam.gserviceaccount.com
    client_id: "123456789737813416734"
    auth_uri: https://accounts.google.com/o/oauth2/auth
    token_uri: https://oauth2.googleapis.com/token
    auth_provider_x509_cert_url:
https://www.googleapis.com/oauth2/v1/certs
    client_x509_cert_url:
https://www.googleapis.com/robot/v1/metadata/x509/trident-
sample%40cloud-native-data.iam.gserviceaccount.com
  credentials:
    name: backend-tbc-gcnv-secret
```

▼ **Configuração com filtro StoragePools**

```
---
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-gcnv-secret
type: Opaque
stringData:
  private_key_id: "<id_value>"
  private_key: |
    -----BEGIN PRIVATE KEY-----
    <key_value>
    -----END PRIVATE KEY-----
---

---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-gcnv
spec:
  version: 1
```

```
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: "123455380079"
  location: europe-west6
  serviceLevel: premium
  storagePools:
    - premium-pool1-europe-west6
    - premium-pool2-europe-west6
  apiKey:
    type: service_account
    project_id: my-gcnv-project
    client_email: myproject-prod@my-gcnv-project.iam.gserviceaccount.com
    client_id: "103346282737811234567"
    auth_uri: https://accounts.google.com/o/oauth2/auth
    token_uri: https://oauth2.googleapis.com/token
    auth_provider_x509_cert_url:
https://www.googleapis.com/oauth2/v1/certs
    client_x509_cert_url:
https://www.googleapis.com/robot/v1/metadata/x509/myproject-prod%40my-
gcnv-project.iam.gserviceaccount.com
  credentials:
    name: backend-tbc-gcnv-secret
```

▼ **Configuração do pool virtual**

Essa configuração de back-end define vários pools virtuais em um único arquivo. Os pools virtuais são definidos na `storage` seção. Elas são úteis quando você tem vários pools de storage com suporte a diferentes níveis de serviço e deseja criar classes de storage no Kubernetes que os representem. Rótulos de pool virtual são usados para diferenciar os pools. Por exemplo, no exemplo abaixo `performance` label e `serviceLevel` type é usado para diferenciar pools virtuais.

Você também pode definir alguns valores padrão para serem aplicáveis a todos os pools virtuais e substituir os valores padrão para pools virtuais individuais. No exemplo a seguir, `snapshotReserve` e `exportRule` serve como padrão para todos os pools virtuais.

Para obter mais informações, "Pools virtuais"consulte .

```
---
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-gcnv-secret
type: Opaque
stringData:
  private_key_id: "<id_value>"
  private_key: |
    -----BEGIN PRIVATE KEY-----
    <key_value>
    -----END PRIVATE KEY-----
```

```
---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-gcnv
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: "123455380079"
  location: europe-west6
  apiKey:
    type: service_account
    project_id: my-gcnv-project
    client_email: myproject-prod@my-gcnv-project.iam.gserviceaccount.com
    client_id: "103346282737811234567"
    auth_uri: https://accounts.google.com/o/oauth2/auth
    token_uri: https://oauth2.googleapis.com/token
    auth_provider_x509_cert_url:
https://www.googleapis.com/oauth2/v1/certs
    client_x509_cert_url:
https://www.googleapis.com/robot/v1/metadata/x509/myproject-prod%40my-
gcnv-project.iam.gserviceaccount.com
  credentials:
    name: backend-tbc-gcnv-secret
  defaults:
    snapshotReserve: "10"
    exportRule: 10.0.0.0/24
  storage:
    - labels:
        performance: extreme
      serviceLevel: extreme
      defaults:
        snapshotReserve: "5"
        exportRule: 0.0.0.0/0
    - labels:
        performance: premium
      serviceLevel: premium
    - labels:
        performance: standard
      serviceLevel: standard
```

▼ **Identidade de nuvem para GKE**

Ao instalar Trident com uma identidade de nuvem do GKE, os backends autenticam-se como uma identidade de carga de trabalho e omitem o `apiKey` bloco. A identidade de nuvem é configurada da mesma forma para ambos os drivers do Google Cloud NetApp Volumes. A única diferença é o

`storageDriverName` valor: use `google-cloud-netapp-volumes` para volumes NAS (NFS ou SMB) e `google-cloud-netapp-volumes-san` para volumes SAN (iSCSI block).

> O suporte à identidade na nuvem para o `google-cloud-netapp-volumes-san` driver requer Trident 26.06 ou posterior. Os volumes SAN são provisionados a partir de pools de storage Flex Unified, então defina `serviceLevel` como `flex`.

Driver NAS (<code>google-cloud-netapp-volumes</code>)

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-gcp-gcnv
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: '012345678901'
  network: gcnv-network
  location: us-west2
  serviceLevel: Premium
  storagePool: pool-premium1
```

Driver SAN (<code>google-cloud-netapp-volumes-san</code>)

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-gcp-gcnv-san
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes-san
  projectNumber: '012345678901'
  network: gcnv-network
  location: us-west2
  serviceLevel: flex
  storagePool: pool-flex1
```

▼ **Configuração de topologias compatíveis**

O Trident facilita o provisionamento de volumes para workloads com base em regiões e zonas de disponibilidade. O `supportedTopologies` bloco nesta configuração de back-end é usado para fornecer uma lista de regiões e zonas por back-end. Os valores de região e zona especificados aqui devem corresponder aos valores de região e zona dos rótulos em cada nó de cluster do Kubernetes. Essas regiões e zonas representam a lista de valores permitidos que podem ser fornecidos em uma classe de armazenamento. Para classes de armazenamento que contêm um subconjunto das regiões e zonas fornecidas em um back-end, o Trident cria volumes na região e na zona mencionadas. Para obter mais informações, "Use a topologia CSI"consulte .

```
---
version: 1
storageDriverName: google-cloud-netapp-volumes
subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
tenantID: 68e4f836-edc1-fake-bff9-b2d865ee56cf
clientID: dd043f63-bf8e-fake-8076-8de91e5713aa
clientSecret: SECRET
location: asia-east1
serviceLevel: flex
supportedTopologies:
  - topology.kubernetes.io/region: asia-east1
    topology.kubernetes.io/zone: asia-east1-a
  - topology.kubernetes.io/region: asia-east1
    topology.kubernetes.io/zone: asia-east1-b
```

O que se segue?

Depois de criar o arquivo de configuração de back-end, execute o seguinte comando:

```
kubectl create -f <backend-file>
```

Para verificar se o back-end foi criado com sucesso, execute o seguinte comando:

```
kubectl get tridentbackendconfig

NAME                BACKEND NAME        BACKEND UUID
PHASE   STATUS
backend-tbc-gcnv    backend-tbc-gcnv    b2fd1ff9-b234-477e-88fd-713913294f65
Bound   Success
```

Se a criação do backend falhar, algo está errado com a configuração do backend. Você pode descrever o back-end usando o `kubectl get tridentbackendconfig <backend-name>` comando ou visualizar os logs para determinar a causa executando o seguinte comando:

```
tridentctl logs
```

Depois de identificar e corrigir o problema com o arquivo de configuração, você pode excluir o backend e executar o comando create novamente.

Definições de classe de armazenamento

A seguir está uma definição básica `StorageClass` que se refere ao backend acima.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gcnv-nfs-sc
provisioner: csi.trident.netapp.io
parameters:
  backendType: "google-cloud-netapp-volumes"
```

- Exemplo de definições usando o `parameter.selector` campo:*

Usando `parameter.selector` você pode especificar para cada `StorageClass` um "pool virtual" que é usado para hospedar um volume. O volume terá os aspetos definidos no pool escolhido.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: extreme-sc
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=extreme
  backendType: google-cloud-netapp-volumes

---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: premium-sc
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=premium
  backendType: google-cloud-netapp-volumes

---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: standard-sc
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=standard
  backendType: google-cloud-netapp-volumes
```

Para obter mais detalhes sobre classes de armazenamento, "Crie uma classe de armazenamento"consulte .

Definições de exemplo para volumes SMB

Usando `nasType`, `node-stage-secret-name` e `node-stage-secret-namespace`, você pode especificar um volume SMB e fornecer as credenciais necessárias do ative Directory. Qualquer usuário/senha do ative Directory com nenhuma ou nenhuma permissão pode ser usada para o segredo da etapa do nó.

▼ **Configuração básica no namespace padrão**

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gcnv-sc-smb
provisioner: csi.trident.netapp.io
parameters:
  backendType: "google-cloud-netapp-volumes"
  trident.netapp.io/nasType: "smb"
  csi.storage.k8s.io/node-stage-secret-name: "smbcreds"
  csi.storage.k8s.io/node-stage-secret-namespace: "default"
```

▼ **Usando diferentes segredos por namespace**

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gcnv-sc-smb
provisioner: csi.trident.netapp.io
parameters:
  backendType: "google-cloud-netapp-volumes"
  trident.netapp.io/nasType: "smb"
  csi.storage.k8s.io/node-stage-secret-name: "smbcreds"
  csi.storage.k8s.io/node-stage-secret-namespace: ${pvc.namespace}
```

▼ **Usando diferentes segredos por volume**

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gcnv-sc-smb
provisioner: csi.trident.netapp.io
parameters:
  backendType: "google-cloud-netapp-volumes"
  trident.netapp.io/nasType: "smb"
  csi.storage.k8s.io/node-stage-secret-name: ${pvc.name}
  csi.storage.k8s.io/node-stage-secret-namespace: ${pvc.namespace}
```

`nasType: smb` Filtros para pools compatíveis com volumes SMB. `nasType: nfs` Ou

`nasType: null` filtros para NFS Pools.

Exemplo de definição de PVC

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: gcnv-nfs-pvc
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 100Gi
  storageClassName: gcnv-nfs-sc
```

Para verificar se o PVC está vinculado, execute o seguinte comando:

```
kubectl get pvc gcnv-nfs-pvc

NAME          STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS AGE
gcnv-nfs-pvc  Bound    pvc-b00f2414-e229-40e6-9b16-ee03eb79a213  100Gi
RWX            gcnv-nfs-sc  1m
```

Configurar o auto-tiering para Google Cloud NetApp Volumes

O auto-tiering é configurado por meio dos parâmetros do backend do Trident e das anotações PersistentVolumeClaim durante o provisionamento de volumes. Você pode configurar o auto-tiering para Google Cloud NetApp Volumes usando Trident.

Visão geral

O recurso de tiering automático permite que o Trident provisione volumes que movem automaticamente dados inativos de uma camada de desempenho para uma camada de capacidade. Isso reduz o custo de armazenamento enquanto preserva o desempenho para dados acessados com frequência.

Trident aplica as configurações de armazenamento em camadas automático somente no momento da criação do volume. Alterações posteriores ao provisionamento não são suportadas em Trident 26.10.

Conceitos

Tiering automático

O auto-tiering move dados acessados com pouca frequência de uma camada de desempenho para uma camada de capacidade com base em padrões de acesso. A movimentação de dados ocorre de forma assíncrona e não é imediata.

Política de tiering

A política de tiering determina se o auto-tiering está habilitado para um volume.

As seguintes políticas são suportadas: * `auto`: ativa o tiering automático com base em padrões de acesso * `none`: desativa o tiering automático

Dias de resfriamento

Os dias de resfriamento especificam o número mínimo de dias que um bloco de dados deve permanecer inativo antes de se tornar elegível para o armazenamento em camadas. Os dias de resfriamento se aplicam somente quando a política de armazenamento em camadas está definida como `auto`.

Modelo de configuração

Escopos de configuração

O auto-tiering pode ser configurado em vários escopos:

- **Escopo do pool de armazenamento** Aplica-se a todos os volumes provisionados do pool.
- **Escopo do volume** Aplica-se a um único volume por meio de anotações PersistentVolumeClaim.

Trident determina a configuração efetiva com base em onde cada configuração está definida.

Precedência de configuração

Quando a mesma configuração é definida em vários escopos, Trident aplica a seguinte ordem de precedência:

1. Anotações de PersistentVolumeClaim
2. Configuração do backend Trident
3. Padrões do pool de armazenamento

As configurações definidas em uma precedência mais alta substituem os valores de nível inferior.

Funcionalidade suportada no Trident 26.10

Trident 26.10 oferece suporte aos seguintes recursos de auto-tiering para Google Cloud NetApp Volumes:

- Habilitar ou desabilitar o auto-tiering durante o provisionamento de volumes
- Definindo uma política de hierarquização na configuração do backend do Trident
- Substituindo a política de escalonamento e os dias de resfriamento por volume usando anotações de PVC
- Configurando dias de resfriamento para volumes com auto-tiering ativado

Funcionalidade não suportada no Trident 26.10

As seguintes operações não são suportadas:

- Modificando as configurações de auto-tiering após a criação do volume
- Alterando políticas de camadas em volumes existentes usando atualizações do Kubernetes
- Aplicando configurações de armazenamento em camadas automático fora dos fluxos de trabalho de provisionamento gerenciados pelo Trident

Parâmetros de configuração do backend

Os seguintes parâmetros controlam o comportamento de auto-tiering quando definidos na configuração do backend Trident:

Parâmetro	Obrigatório	Descrição
tieringPolicy	Não	Política de escalonamento para volumes (`auto` ou `none`)
tieringMinimumCoolingDays	Não	Número de dias de inatividade antes dos dados serem transferidos de camada (intervalo: 2–183, padrão: 31)

Substituições em nível de volume usando PersistentVolumeClaim anotações

Anotações suportadas

PersistentVolumeClaim anotações permitem a substituição das configurações de auto-tiering por volume.

Anotação	Descrição
`trident.netapp.io/tieringPolicy`	Substitui a política de hierarquização para o volume
`trident.netapp.io/tieringMinimumCoolingDays`	Substitui o valor dos dias de resfriamento para o volume

Exemplo: PersistentVolumeClaim com substituições de hierarquização automática

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: auto-tiering-pvc
  annotations:
    trident.netapp.io/tieringPolicy: auto
    trident.netapp.io/tieringMinimumCoolingDays: "45"
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: google-cloud-netapp-volumes-auto-tiering
  resources:
    requests:
      storage: 500Gi
```

Comportamento e limitações

Comportamento de provisionamento

- As configurações de auto-tiering são avaliadas e aplicadas somente no momento da criação do volume.
- Trident não reconcilia a configuração de tiering após o provisionamento.

- Os dias de resfriamento são ignorados quando a política de tiering está definida como `none`.

Limitações da plataforma

- O auto-tiering é compatível apenas com volumes NAS (NFS e SMB).
- Volumes em bloco (iSCSI) não suportam auto-tiering.
- O pool de storage do Google Cloud NetApp Volumes deve ter o armazenamento em camadas automático ativado no Google Cloud.

Valores suportados

- Intervalo válido para `tieringMinimumCoolingDays`: 2 a 183
- Valor padrão: 31

Configurar pools de storage do Google Cloud NetApp Volumes em modo ONTAP

A partir do Trident 26.06, você pode configurar `ontap-san` e `ontap-nas` back-ends existentes para provisionar armazenamento em pools de storage do Google Cloud NetApp Volumes (GCNV) no modo ONTAP. Este recurso é uma Tech Preview.

OBSERVAÇÃO	Este recurso é uma Prévia Técnica no Trident 26.10. Recursos em Tech Preview não são suportados para uso em produção. A funcionalidade, os campos de configuração e a matriz de suporte podem ser alterados antes da disponibilidade geral (GA).

Como funciona

Trident reutiliza a `ontap-san` e `ontap-nas` lógica de driver existente e encaminha as chamadas REST do ONTAP por meio de um endpoint proxy do GCNV. Essa abordagem permite que você provisione e gerencie armazenamento em clusters ONTAP hospedados pelo GCNV, mantendo os mesmos fluxos de trabalho operacionais que você usa para um backend ONTAP direto. O modo ONTAP não introduz um novo driver de armazenamento. Você habilita isso por backend adicionando um `gcnv` bloco de configuração a um `ontap-san` ou `ontap-nas` backend.

O caminho de provisionamento é o seguinte:

`PersistentVolumeClaim`→ Trident (`ontap-san` ou `ontap-nas`) → ONTAP cliente → GCNV proxy → ONTAP cluster no pool GCNV

Escopo suportado e não suportado

Esta versão de pré-visualização técnica oferece suporte ao seguinte:

- Drivers: `ontap-san` (iSCSI) e `ontap-nas` (NFS ou SMB).
- Operações de ciclo de vida por meio do proxy: inicialização do backend, criação e exclusão de volumes e fluxos REST do ONTAP relacionados.
- Modelos de autenticação: Workload Identity Pool (WIP), chave de conta de serviço e fallback de Application Default Credentials (ADC).

Os seguintes itens estão fora do escopo desta Tech Preview:

- Os `ontap-san-economy`, `ontap-nas-economy` e `ontap-nas-flexgroup` drivers.
- O caminho de personalidade ASA r2.
- Recurso alternativo ZAPI. O modo ONTAP utiliza apenas o ONTAP REST.

Pré-requisitos

Antes de configurar um backend no modo ONTAP, certifique-se de ter o seguinte:

- Trident 26.06 ou posterior.
- Um pool de storage GCNV ONTAP-mode em seu projeto e localização de destino.
- Permissões do GCP IAM para operações de proxy, com escopo adequado ao seu ambiente.
- Uma configuração de backend que inclui `proxyURL`, `projectNumber`, `location` e `poolID`.
- Para fluxos de trabalho iSCSI SAN, os pré-requisitos de iSCSI e multipath do lado do nó para sua plataforma. Consulte "Prepare o nó de trabalho".

Configurar o backend

É `TridentBackendConfig` necessário `credentials` referenciar um Secret do Kubernetes. Para o modo ONTAP, o Secret fornece credenciais para acesso ao proxy do GCP, não ao ONTAP `managementLIF` / `username` / `password`. Dependendo do método de autenticação, ele pode conter campos de chave da conta de serviço ou estar vazio. `svm` é opcional: cada pool do GCNV no modo ONTAP atualmente tem uma SVM, e o Trident a deriva do pool quando ela é omitida.

Para a autenticação de chave da conta de serviço, use o mesmo padrão do GCNV nativo: campos não sensíveis em `gcnv.apiKey`, `private_key` e `private_key_id` no secret. Consulte "Exemplos de configuração do Google Cloud NetApp Volumes". Para a autenticação do Workload Identity Pool ou da identidade de nuvem do GKE, consulte os exemplos em Autenticação.

backend ontap-san

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: gcnv-ontap-san
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-san
  backendName: gcnv-ontap-san
  credentials:
    name: gcnv-sa-secret
    type: secret
  gcnv:
    proxyURL: "https://netapp.googleapis.com"
    projectNumber: "<project-number>"
    location: "<pool-location>"
    poolID: "<pool-id>"
```

```
  svm: "<svm-name>"
```

backend ontap-nas

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: gcnv-ontap-nas
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-nas
  backendName: gcnv-ontap-nas
  credentials:
    name: gcnv-sa-secret
    type: secret
  gcnv:
    proxyURL: "https://netapp.googleapis.com"
    projectNumber: "<project-number>"
    location: "<pool-location>"
    poolID: "<pool-id>"
    apiKey:
      type: service_account
      project_id: "<project-id>"
      client_email: "<service-account-email>"
      client_id: "<client-id>"
      auth_uri: "https://accounts.google.com/o/oauth2/auth"
      token_uri: "https://oauth2.googleapis.com/token"
      auth_provider_x509_cert_url:
"https://www.googleapis.com/oauth2/v1/certs"
      client_x509_cert_url: "<client-x509-cert-url>"
  svm: "<svm-name>"
```

Segredo

```
apiVersion: v1
kind: Secret
metadata:
  name: gcnv-sa-secret
  namespace: trident
type: Opaque
stringData:
  private_key_id: "<private-key-id>"
  private_key: |
    -----BEGIN PRIVATE KEY-----
```

```
    <private-key>
    -----END PRIVATE KEY-----
```

OBSERVAÇÃO	Não coloque `private_key` ou `private_key_id` na `TridentBackendConfig` especificação — o CRD os rejeita. Use `ontap-san` com o mesmo `gcnv` e padrão de credencial para pool de storage em bloco.

Autenticação

O Trident resolve as credenciais para acesso por proxy na seguinte ordem:

1. Pool de identidade de workload (WIP)
2. Chave da conta de serviço
3. Credenciais padrão do aplicativo (ADC)

Para esta prévia técnica, utilize as seguintes práticas:

- Dê preferência ao Workload Identity Pool quando ele estiver disponível.
- Não incorpore chaves privadas brutas em arquivos de backend com controle de versão. Armazene as credenciais em um segredo do Kubernetes.
- Aplique o escopo de IAM de menor privilégio à conta de serviço.

O `location` no `gcnv` bloco deve corresponder à localização do pool: use a zona para um pool zonal e a região para um pool regional.

O Workload Identity Pool

Use `gcnv.wipCredential` para federação explícita de conta externa configurada no `TridentBackendConfig`. No OpenShift, o Trident projeta o token em `/var/run/secrets/wif-token/token`; este é o único caminho de token que o Trident projeta para o OpenShift WIF, portanto use-o para `credential_source.file`. Instale o Trident com a audience do WIF, conforme descrito em "Configure o Google Cloud NetApp Volumes com a federação de identidade da carga de trabalho". O `credentials` campo ainda faz referência a um Secret, mas o Secret não deve conter um `private_key`.

```
apiVersion: v1
kind: Secret
metadata:
  name: gcnv-ontap-empty
  namespace: trident
type: Opaque
---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: gcnv-ontap-nas-wip
  namespace: trident
spec:
```

```
  version: 1
  storageDriverName: ontap-nas
  backendName: gcnv-ontap-nas-wip
  useREST: true
  nasType: nfs
  nfsMountOptions: nfsvers=4.1
  credentials:
    name: gcnv-ontap-empty
    type: secret
  gcnv:
    proxyURL: "https://netapp.googleapis.com"
    projectNumber: "<project-number>"
    location: "<pool-location>"
    poolID: "<pool-id>"
    wipCredential:
      type: external_account
      audience: "//iam.googleapis.com/projects/<wif-pool-project-
number>/locations/global/workloadIdentityPools/<wif-pool-
id>/providers/<provider-id>"
      subject_token_type: "urn:ietf:params:oauth:token-type:jwt"
      token_url: "https://sts.googleapis.com/v1/token"
      service_account_impersonation_url:
"https://iamcredentials.googleapis.com/v1/projects/-
/serviceAccounts/<service-account-email>:generateAccessToken"
      credential_source:
        file: "/var/run/secrets/wif-token/token"
```

Identidade de nuvem do GKE

Para o GKE, configure a identidade de nuvem do Trident durante a instalação, conforme descrito em "Usar a identidade de carga de trabalho do GKE". O backend usa Application Default Credentials (ADC); omita `gcnv.apiKey` e `gcnv.wipCredential`. O campo `credentials` ainda faz referência a um Secret, mas não coloque um `private_key` nele.

```
apiVersion: v1
kind: Secret
metadata:
  name: gcnv-ontap-empty
  namespace: trident
type: Opaque
---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: gcnv-ontap-nas-wi
  namespace: trident
```

```
spec:
  version: 1
  storageDriverName: ontap-nas
  backendName: gcnv-ontap-nas-wi
  useREST: true
  nasType: nfs
  nfsMountOptions: nfsvers=4.1
  credentials:
    name: gcnv-ontap-empty
    type: secret
  gcnv:
    proxyURL: "https://netapp.googleapis.com"
    projectNumber: "<project-number>"
    location: "<pool-location>"
    poolID: "<pool-id>"
```

Comportamento de validação e inicialização

O Trident valida um backend no modo ONTAP durante a inicialização e falha rapidamente em vez de inicializar parcialmente.

A inicialização do backend falha se um `gcnv`campo obrigatório estiver ausente, se o driver não for compatível com o modo ONTAP ou se a resolução de credenciais falhar.

Limitações conhecidas

- Este recurso é uma Prévia técnica. O comportamento e a matriz suportada podem mudar antes do GA.
- O fallback do ZAPI não é usado no modo ONTAP.
- Os requisitos de protocolo e de prontidão do nó ainda se aplicam. Por exemplo, os fluxos de trabalho do iSCSI exigem os pré-requisitos do lado do nó descritos em "Prepare o nó de trabalho".
- As restrições existentes dos ambientes ONTAP e GCNV ainda se aplicam.

Upgrade e compatibilidade

O modo ONTAP é opcional para cada backend por meio do `gcnv` bloco de configuração:

- Os backends ONTAP existentes que não são GCNV não são afetados.
- Implantações mistas que combinam backends diretos do ONTAP e backends de proxy no modo ONTAP são compatíveis por meio da configuração no nível do backend. Cada backend é configurado independentemente.
- Para parar de usar o ONTAP-mode, remova ou substitua os backends afetados. Nenhuma opção global é necessária.

O que se segue?

Aplique o segredo antes do TBC; use `kubectl apply` e `-n trident` para corresponder aos exemplos.

Configurar um back-end NetApp HCI ou SolidFire

Saiba como criar e usar um backend Element com a instalação do Trident.

Detalhes do driver do elemento

O Trident fornece ao `solidfire-san` controlador de armazenamento a comunicação com o cluster. Os modos de acesso suportados são: *ReadWriteOnce* (RWO), *ReadOnlyMuy* (ROX), *ReadWriteMany* (RWX), *ReadWriteOncePod* (RWOP).

O `solidfire-san` driver de armazenamento suporta os modos de volume *file* e *block*. Para o `Filesystem` volumeMode, o Trident cria um volume e cria um sistema de arquivos. O tipo de sistema de arquivos é especificado pelo StorageClass.

Condutor	Protocolo	Modo de volume	Modos de acesso suportados	Sistemas de arquivos suportados
`solidfire-san`	ISCSI	Bloco	RWO, ROX, RWX, RWOP	Sem sistema de ficheiros. Dispositivo de bloco bruto.
`solidfire-san`	ISCSI	Sistema de ficheiros	RWO, RWOP	`xfs` `ext3`, , `ext4`

Antes de começar

Você precisará do seguinte antes de criar um backend de elemento.

- Um sistema de storage compatível que executa o software Element.
- Credenciais para um usuário de administrador ou locatário de cluster do NetApp HCI/SolidFire que possa gerenciar volumes.
- Todos os seus nós de trabalho do Kubernetes devem ter as ferramentas iSCSI apropriadas instaladas. "informações sobre a preparação do nó de trabalho"Consulte a .

Opções de configuração de back-end

Consulte a tabela a seguir para obter as opções de configuração de back-end:

Parâmetro	Descrição	Padrão
`version`		Sempre 1
`storageDriverName`	Nome do controlador de armazenamento	Sempre "SolidFire-san"
`backendName`	Nome personalizado ou back-end de storage	Endereço IP "SolidFire_" e armazenamento (iSCSI)
`Endpoint`	MVIP para o cluster SolidFire com credenciais de locatário	
`SVIP`	Porta e endereço IP de armazenamento (iSCSI)	

Parâmetro	Descrição	Padrão
`labels`	Conjunto de rótulos arbitrários formatados em JSON para aplicar em volumes.	""
`TenantName`	Nome do locatário a utilizar (criado se não for encontrado)	
`InitiatorIFace`	Restringir o tráfego iSCSI a uma interface de host específica	"predefinição"
`UseCHAP`	Use CHAP para autenticar iSCSI. Trident usa CHAP.	verdadeiro
`AccessGroups`	Lista de IDs de Grupo de Acesso a utilizar	Encontra a ID de um grupo de acesso chamado "Trident"
`Types`	Especificações de QoS	
`limitVolumeSize`	Falha no provisionamento se o tamanho do volume solicitado estiver acima desse valor	"" (não aplicado por padrão)
`debugTraceFlags`	Debug flags para usar ao solucionar problemas. Por exemplo, "api":false, "method":true"	nulo

AVISO Não use `debugTraceFlags` a menos que você esteja solucionando problemas e exija um despejo de log detalhado.

Exemplo 1: Configuração de back-end para `solidfire-san` driver com três tipos de volume

Este exemplo mostra um arquivo de back-end usando autenticação CHAP e modelagem de três tipos de volume com garantias de QoS específicas. Provavelmente você definiria classes de armazenamento para consumir cada uma delas usando o `IOPS` parâmetro de classe de armazenamento.

```
---
version: 1
storageDriverName: solidfire-san
Endpoint: https://<user>:<password>@<mvip>/json-rpc/8.0
SVIP: <svip>:3260
TenantName: <tenant>
labels:
  k8scluster: dev1
  backend: dev1-element-cluster
UseCHAP: true
Types:
  - Type: Bronze
    Qos:
      minIOPS: 1000
      maxIOPS: 2000
      burstIOPS: 4000
```

```
  - Type: Silver
    Qos:
      minIOPS: 4000
      maxIOPS: 6000
      burstIOPS: 8000
  - Type: Gold
    Qos:
      minIOPS: 6000
      maxIOPS: 8000
      burstIOPS: 10000
```

Exemplo 2: Configuração de classe de back-end e armazenamento para `solidfire-san` driver com pools virtuais

Este exemplo mostra o arquivo de definição de back-end configurado com pools virtuais junto com o StorageClasses que se referem a eles.

O Trident copia rótulos presentes em um pool de storage para a LUN de storage de back-end no provisionamento. Por conveniência, os administradores de storage podem definir rótulos por pool virtual e volumes de grupo por rótulo.

No arquivo de definição de back-end de exemplo mostrado abaixo, padrões específicos são definidos para todos os pools de armazenamento, que definem o `type` em Prata. Os pools virtuais são definidos na `storage` seção. Neste exemplo, alguns dos pools de armazenamento definem seu próprio tipo, e alguns pools substituem os valores padrão definidos acima.

```
---
version: 1
storageDriverName: solidfire-san
Endpoint: https://<user>:<password>@<mvip>/json-rpc/8.0
SVIP: <svip>:3260
TenantName: <tenant>
UseCHAP: true
Types:
  - Type: Bronze
    Qos:
      minIOPS: 1000
      maxIOPS: 2000
      burstIOPS: 4000
  - Type: Silver
    Qos:
      minIOPS: 4000
      maxIOPS: 6000
      burstIOPS: 8000
  - Type: Gold
    Qos:
      minIOPS: 6000
```

```
      maxIOPS: 8000
      burstIOPS: 10000
type: Silver
labels:
  store: solidfire
  k8scluster: dev-1-cluster
region: us-east-1
storage:
  - labels:
      performance: gold
      cost: "4"
    zone: us-east-1a
    type: Gold
  - labels:
      performance: silver
      cost: "3"
    zone: us-east-1b
    type: Silver
  - labels:
      performance: bronze
      cost: "2"
    zone: us-east-1c
    type: Bronze
  - labels:
      performance: silver
      cost: "1"
    zone: us-east-1d
```

As seguintes definições do StorageClass referem-se aos pools virtuais acima. Usando o `parameters.selector` campo, cada StorageClass chama qual(s) pool(s) virtual(s) pode(m) ser(ão) usado(s) para hospedar um volume. O volume terá os aspetos definidos no pool virtual escolhido.

O primeiro StorageClass) (`solidfire-gold-four`será mapeado para o primeiro pool virtual. Este é o único pool que oferece desempenho de ouro com um `Volume Type QoS` de ouro. O último StorageClass) (`solidfire-silver`chama qualquer pool de armazenamento que ofereça um desempenho prateado. O Trident decidirá qual pool virtual é selecionado e garante que o requisito de armazenamento seja atendido.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: solidfire-gold-four
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=gold; cost=4
  fsType: ext4
```

```
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: solidfire-silver-three
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=silver; cost=3
  fsType: ext4

---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: solidfire-bronze-two
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=bronze; cost=2
  fsType: ext4

---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: solidfire-silver-one
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=silver; cost=1
  fsType: ext4

---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: solidfire-silver
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=silver
  fsType: ext4
```

Encontre mais informações

- "Grupos de acesso de volume"

Controladores SAN ONTAP

Descrição geral do controlador SAN ONTAP

Saiba mais sobre como configurar um back-end ONTAP com drivers SAN ONTAP e Cloud Volumes ONTAP.

Detalhes do driver SAN ONTAP

O Trident fornece os seguintes drivers de armazenamento SAN para se comunicar com o cluster ONTAP. Os modos de acesso suportados são: *ReadWriteOnce* (RWO), *ReadOnlyMuy* (ROX), *ReadWriteMany* (RWX), *ReadWriteOncePod* (RWOP).

Condutor	Protocolo	VolumeMode	Modos de acesso suportados	Sistemas de arquivos suportados
`ontap-san`	SCSI iSCSI em FC	Bloco	RWO, ROX, RWX, RWOP	Sem sistema de arquivos; dispositivo de bloco bruto
`ontap-san`	SCSI iSCSI em FC	Sistema de ficheiros	RWO, RWOP ROX e RWX não estão disponíveis no modo de volume do sistema de arquivos.	`xfs` `ext3`, , `ext4`
`ontap-san`	NVMe/TCP Considerações adicionais para NVMe/TCP Consulte a .	Bloco	RWO, ROX, RWX, RWOP	Sem sistema de arquivos; dispositivo de bloco bruto
`ontap-san`	NVMe/TCP Considerações adicionais para NVMe/TCP Consulte a .	Sistema de ficheiros	RWO, RWOP ROX e RWX não estão disponíveis no modo de volume do sistema de arquivos.	`xfs` `ext3`, , `ext4`
`ontap-san-economy`	ISCSI	Bloco	RWO, ROX, RWX, RWOP	Sem sistema de arquivos; dispositivo de bloco bruto
`ontap-san-economy`	ISCSI	Sistema de ficheiros	RWO, RWOP ROX e RWX não estão disponíveis no modo de volume do sistema de arquivos.	`xfs` `ext3`, , `ext4`

AVISO

- Use `ontap-san-economy` somente se a contagem de uso de volume persistente for esperada ser maior que "Limites de volume ONTAP suportados".

- Use `ontap-nas-economy` somente se a contagem de uso de volume persistente for esperada para ser maior do que "Limites de volume ONTAP suportados" e o `ontap-san-economy` driver não puder ser usado.
- Não use o uso `ontap-nas-economy` se você antecipar a necessidade de proteção de dados, recuperação de desastres ou mobilidade.
- O dimensionamento automático de volume do ONTAP é compatível com `ontap-san` e `ontap-san-economy`. Não é compatível com os drivers NAS do ONTAP. Para obter detalhes e limitações, consulte "Configurar o dimensionamento automático de volume do ONTAP".

Permissões do usuário

O Trident espera ser executado como administrador do ONTAP ou SVM, normalmente usando o `admin` usuário do cluster ou um `vsadmin` usuário SVM, ou um usuário com um nome diferente que tenha a mesma função. Para implantações do Amazon FSX for NetApp ONTAP, o Trident espera ser executado como administrador do ONTAP ou SVM, usando o usuário do cluster `fsxadmin` ou um `vsadmin` usuário SVM, ou um usuário com um nome diferente que tenha a mesma função. O `fsxadmin` usuário é um substituto limitado para o usuário administrador do cluster.

OBSERVAÇÃO	Se você usar o `limitAggregateUsage` parâmetro, as permissões de administrador do cluster serão necessárias. Ao usar o Amazon FSX for NetApp ONTAP com Trident, o `limitAggregateUsage` parâmetro não funcionará com as `vsadmin` contas de usuário e `fsxadmin`. A operação de configuração falhará se você especificar este parâmetro.

Embora seja possível criar uma função mais restritiva no ONTAP que um driver Trident pode usar, não recomendamos. A maioria das novas versões do Trident chamarão APIs adicionais que teriam que ser contabilizadas, tornando as atualizações difíceis e suscetíveis a erros.

Considerações adicionais para NVMe/TCP

O Trident dá suporte ao protocolo NVMe (non-volátil Memory Express) usando `ontap-san` o driver, incluindo:

- IPv6
- Snapshots e clones de volumes NVMe
- Redimensionamento de um volume NVMe
- Importação de um volume NVMe que foi criado fora do Trident para que seu ciclo de vida possa ser gerenciado pelo Trident
- Multipathing nativo NVMe
- Desligamento gracioso ou vergonhoso dos K8s nós (24,06)

O Trident não suporta:

- DH-HMAC-CHAP que é suportado nativamente por NVMe
- Multipathing de mapeador de dispositivos (DM)
- Criptografia do Linux Unified Key Setup (LUKS)

OBSERVAÇÃO	O NVMe é compatível apenas com APIs REST ONTAP e não com ONTAPI (ZAPI).

Prepare-se para configurar o back-end com drivers SAN ONTAP

Entenda os requisitos e as opções de autenticação para configurar um back-end do ONTAP com drivers de SAN ONTAP.

Requisitos

Para todos os backends ONTAP, o Trident exige que pelo menos um agregado seja atribuído ao SVM.

OBSERVAÇÃO	"Sistemas ASA r2"Diferem de outros sistemas ONTAP (ASA, AFF e FAS) na implementação de sua camada de armazenamento. Nos sistemas ASA r2, as zonas de disponibilidade de armazenamento são usadas em vez de agregados. Consulte"isto" Artigo da Base de Conhecimento sobre como atribuir agregados a SVMs em sistemas ASA r2.

Lembre-se de que você também pode executar mais de um driver e criar classes de armazenamento que apontam para um ou outro. Por exemplo, você pode configurar uma `san-dev` classe que usa o `ontap-san` driver e uma `san-default` classe que usa a `ontap-san-economy` mesma.

Todos os seus nós de trabalho do Kubernetes devem ter as ferramentas iSCSI apropriadas instaladas. "Prepare o nó de trabalho"Consulte para obter detalhes.

Autenticar o back-end do ONTAP

O Trident oferece dois modos de autenticar um back-end do ONTAP.

- Baseado em credenciais: O nome de usuário e senha para um usuário do ONTAP com as permissões necessárias. Recomenda-se a utilização de uma função de início de sessão de segurança predefinida, como `admin` ou `vsadmin` para garantir a máxima compatibilidade com as versões do ONTAP.
- Baseado em certificado: O Trident também pode se comunicar com um cluster ONTAP usando um certificado instalado no back-end. Aqui, a definição de back-end deve conter valores codificados em Base64 do certificado de cliente, chave e certificado de CA confiável, se usado (recomendado).

Você pode atualizar os backends existentes para mover entre métodos baseados em credenciais e baseados em certificado. No entanto, apenas um método de autenticação é suportado por vez. Para alternar para um método de autenticação diferente, você deve remover o método existente da configuração de back-end.

AVISO	Se você tentar fornecer **credenciais e certificados**, a criação de back-end falhará com um erro que mais de um método de autenticação foi fornecido no arquivo de configuração.

Ative a autenticação baseada em credenciais

O Trident requer as credenciais para um administrador com escopo SVM/escopo de cluster para se comunicar com o back-end do ONTAP. Recomenda-se a utilização de funções padrão predefinidas, como `admin` ou `vsadmin`. Isso garante compatibilidade direta com futuras versões do ONTAP que podem expor APIs de recursos a serem usadas por futuras versões do Trident. Uma função de login de segurança personalizada pode ser criada e usada com o Trident, mas não é recomendada.

Uma definição de backend de exemplo será assim:

YAML

```yaml
---
version: 1
backendName: ExampleBackend
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_nfs
username: vsadmin
password: password
```

JSON

```json
{
  "version": 1,
  "backendName": "ExampleBackend",
  "storageDriverName": "ontap-san",
  "managementLIF": "10.0.0.1",
  "svm": "svm_nfs",
  "username": "vsadmin",
  "password": "password"
}
```

Tenha em mente que a definição de back-end é o único lugar onde as credenciais são armazenadas em texto simples. Depois que o back-end é criado, os nomes de usuário/senhas são codificados com Base64 e armazenados como segredos do Kubernetes. A criação ou atualização de um backend é a única etapa que requer conhecimento das credenciais. Como tal, é uma operação somente de administrador, a ser realizada pelo administrador do Kubernetes/storage.

Habilitar autenticação baseada em certificado

Backends novos e existentes podem usar um certificado e se comunicar com o back-end do ONTAP. Três parâmetros são necessários na definição de backend.

- ClientCertificate: Valor codificado base64 do certificado do cliente.
- ClientPrivateKey: Valor codificado em base64 da chave privada associada.
- TrustedCACertificate: Valor codificado base64 do certificado CA confiável. Se estiver usando uma CA confiável, esse parâmetro deve ser fornecido. Isso pode ser ignorado se nenhuma CA confiável for usada.

Um fluxo de trabalho típico envolve as etapas a seguir.

Passos

1. Gerar um certificado e chave de cliente. Ao gerar, defina Nome Comum (CN) para o usuário ONTAP para autenticar como.

```
openssl req -x509 -nodes -days 1095 -newkey rsa:2048 -keyout k8senv.key
```

```
-out k8senv.pem -subj "/C=US/ST=NC/L=RTP/O=NetApp/CN=admin"
```

2. Adicionar certificado de CA confiável ao cluster do ONTAP. Isso pode já ser Tratado pelo administrador do armazenamento. Ignore se nenhuma CA confiável for usada.

```
security certificate install -type server -cert-name <trusted-ca-cert-
name> -vserver <vserver-name>
ssl modify -vserver <vserver-name> -server-enabled true -client-enabled
true -common-name <common-name> -serial <SN-from-trusted-CA-cert> -ca
<cert-authority>
```

3. Instale o certificado e a chave do cliente (a partir do passo 1) no cluster do ONTAP.

```
security certificate install -type client-ca -cert-name <certificate-
name> -vserver <vserver-name>
security ssl modify -vserver <vserver-name> -client-enabled true
```

OBSERVAÇÃO	Após executar este comando, ONTAP solicitará a entrada do certificado. Cole o conteúdo do `k8senv.pem` arquivo gerado na etapa 1, depois insira `END` para concluir a instalação.

4. Confirme se a função de login de segurança do ONTAP suporta `cert` o método de autenticação.

```
security login create -user-or-group-name admin -application ontapi
-authentication-method cert
security login create -user-or-group-name admin -application http
-authentication-method cert
```

5. Teste a autenticação usando certificado gerado. Substitua o ONTAP Management LIF> e o <vserver name> por IP de LIF de gerenciamento e nome da SVM.

```
curl -X POST -Lk https://<ONTAP-Management-
LIF>/servlets/netapp.servlets.admin.XMLrequest_filer --key k8senv.key
--cert ~/k8senv.pem -d '<?xml version="1.0" encoding="UTF-8"?><netapp
xmlns="http://www.netapp.com/filer/admin" version="1.21"
vfiler="<vserver-name>"><vserver-get></vserver-get></netapp>'
```

6. Codificar certificado, chave e certificado CA confiável com Base64.

```
base64 -w 0 k8senv.pem >> cert_base64
base64 -w 0 k8senv.key >> key_base64
```

```
base64 -w 0 trustedca.pem >> trustedca_base64
```

7. Crie backend usando os valores obtidos na etapa anterior.

```
cat cert-backend.json
{
"version": 1,
"storageDriverName": "ontap-san",
"backendName": "SanBackend",
"managementLIF": "1.2.3.4",
"svm": "vserver_test",
"clientCertificate": "Faaaakkkkeeee...Vaaalllluuuueeee",
"clientPrivateKey": "LS0tFaKE...0VaLuES0tLS0K",
"trustedCACertificate": "QNFinfO...SiqOyN",
"storagePrefix": "myPrefix_"
}

tridentctl create backend -f cert-backend.json -n trident
+------------+----------------+--------------------------------------
+--------+---------+
|    NAME    | STORAGE DRIVER |                 UUID                 |
STATE  | VOLUMES |
+------------+----------------+--------------------------------------
+--------+---------+
| SanBackend | ontap-san      | 586b1cd5-8cf8-428d-a76c-2872713612c1 |
online |       0 |
+------------+----------------+--------------------------------------
+--------+---------+
```

Atualizar métodos de autenticação ou girar credenciais

Você pode atualizar um back-end existente para usar um método de autenticação diferente ou para girar suas credenciais. Isso funciona de ambas as maneiras: Backends que fazem uso de nome de usuário / senha podem ser atualizados para usar certificados; backends que utilizam certificados podem ser atualizados para nome de usuário / senha com base. Para fazer isso, você deve remover o método de autenticação existente e adicionar o novo método de autenticação. Em seguida, use o arquivo backend.json atualizado contendo os parâmetros necessários para executar `tridentctl backend update`.

```
cat cert-backend-updated.json
{
"version": 1,
"storageDriverName": "ontap-san",
"backendName": "SanBackend",
"managementLIF": "1.2.3.4",
"svm": "vserver_test",
```

```
"username": "vsadmin",
"password": "password",
"storagePrefix": "myPrefix_"
}

#Update backend with tridentctl
tridentctl update backend SanBackend -f cert-backend-updated.json -n
trident
+------------+----------------+--------------------------------------
+--------+---------+
|    NAME    | STORAGE DRIVER |                 UUID                 |
STATE  | VOLUMES |
+------------+----------------+--------------------------------------
+--------+---------+
| SanBackend | ontap-san      | 586b1cd5-8cf8-428d-a76c-2872713612c1 |
online |       9 |
+------------+----------------+--------------------------------------
+--------+---------+
```

OBSERVAÇÃO	Ao girar senhas, o administrador de armazenamento deve primeiro atualizar a senha do usuário no ONTAP. Isso é seguido por uma atualização de back-end. Ao girar certificados, vários certificados podem ser adicionados ao usuário. O back-end é então atualizado para usar o novo certificado, seguindo o qual o certificado antigo pode ser excluído do cluster do ONTAP.

A atualização de um back-end não interrompe o acesso a volumes que já foram criados, nem afeta as conexões de volume feitas depois. Uma atualização de back-end bem-sucedida indica que o Trident pode se comunicar com o back-end do ONTAP e lidar com operações de volume futuras.

Crie uma função ONTAP personalizada para o Trident

Você pode criar uma função de cluster do ONTAP com Privileges mínimo para que você não precise usar a função de administrador do ONTAP para executar operações no Trident. Quando você inclui o nome de usuário em uma configuração de back-end do Trident, o Trident usa a função de cluster do ONTAP criada para executar as operações.

"Gerador de função personalizada Trident"Consulte para obter mais informações sobre como criar funções personalizadas do Trident.

Usando a CLI do ONTAP

1. Crie uma nova função usando o seguinte comando:

   ```
   security login role create <role_name\> -cmddirname "command" -access all
   –vserver <svm_name\>
   ```

2. Crie um nome de usuário para o usuário do Trident:

   ```
   security login create -username <user_name\> -application ontapi
   -authmethod <password\> -role <name_of_role_in_step_1\> –vserver
   ```

```
<svm_name\> -comment "user_description"
```

3. Mapeie a função para o usuário:

```
security login modify username <user_name\> –vserver <svm_name\> -role
<role_name\> -application ontapi -application console -authmethod
<password\>
```

Usando o System Manager

Execute as seguintes etapas no Gerenciador do sistema do ONTAP:

1. **Crie uma função personalizada**:
   a. Para criar uma função personalizada no nível do cluster, selecione **Cluster > Settings**.

      (Ou) para criar uma função personalizada no nível SVM, selecione **Storage > Storage VMs > `required SVM` Settings > Users and Roles**.

   b. Selecione o ícone de seta (→) ao lado de **usuários e funções**.
   c. Selecione * Adicionar * em **funções**.
   d. Defina as regras para a função e clique em **Salvar**.
2. **Mapeie a função para o usuário do Trident**: Execute as seguintes etapas na página **usuários e funções**:
   a. Selecione Adicionar ícone * em *usuários.
   b. Selecione o nome de usuário desejado e selecione uma função no menu suspenso para **função**.
   c. Clique em **Salvar**.

Consulte as páginas a seguir para obter mais informações:

- "Funções personalizadas para administração do ONTAP" ou "Definir funções personalizadas"
- "Trabalhe com funções e usuários"

Autenticar conexões com CHAP bidirecional

O Trident pode autenticar sessões iSCSI com CHAP bidirecional para os `ontap-san` drivers e `ontap-san-economy`. Isso requer a ativação da `useCHAP` opção na definição de backend. Quando definido como `true`, o Trident configura a segurança do iniciador padrão do SVM para CHAP bidirecional e define o nome de usuário e os segredos do arquivo de back-end. O NetApp recomenda o uso de CHAP bidirecional para autenticar conexões. Veja a seguinte configuração de exemplo:

```
---
version: 1
storageDriverName: ontap-san
backendName: ontap_san_chap
managementLIF: 192.168.0.135
svm: ontap_iscsi_svm
useCHAP: true
username: vsadmin
```

```
password: password
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSd6cNwxyz
```

AVISO	O `useCHAP` parâmetro é uma opção booleana que pode ser configurada apenas uma vez. Ele é definido como false por padrão. Depois de configurá-lo como verdadeiro, você não pode configurá-lo como falso.

Além `useCHAP=true` do , os `chapInitiatorSecret` campos , `chapTargetInitiatorSecret`, `chapTargetUsername`, e `chapUsername` devem ser incluídos na definição de back-end. Os segredos podem ser alterados depois que um backend é criado executando `tridentctl update`.

Como funciona

Ao definir `useCHAP` como verdadeiro, o administrador de armazenamento instrui o Trident a configurar o CHAP no back-end de armazenamento. Isso inclui o seguinte:

- Configuração do CHAP no SVM:
  - Se o tipo de segurança do iniciador padrão da SVM for nenhum (definido por padrão) **e** não houver LUNs pré-existentes no volume, o Trident definirá o tipo de segurança padrão `CHAP` e continuará configurando o iniciador CHAP e o nome de usuário e os segredos de destino.
  - Se o SVM contiver LUNs, o Trident não ativará o CHAP no SVM. Isso garante que o acesso a LUNs que já estão presentes no SVM não seja restrito.
- Configurando o iniciador CHAP e o nome de usuário e os segredos de destino; essas opções devem ser especificadas na configuração de back-end (como mostrado acima).

Depois que o back-end é criado, o Trident cria um CRD correspondente `tridentbackend` e armazena os segredos e nomes de usuário do CHAP como segredos do Kubernetes. Todos os PVS criados pelo Trident neste backend serão montados e anexados através do CHAP.

Gire as credenciais e atualize os backends

Você pode atualizar as credenciais CHAP atualizando os parâmetros CHAP no `backend.json` arquivo. Isso exigirá a atualização dos segredos CHAP e o uso do `tridentctl update` comando para refletir essas alterações.

AVISO	Ao atualizar os segredos CHAP para um backend, você deve usar `tridentctl` para atualizar o backend. Não atualize as credenciais no cluster de storage usando a CLI da ONTAP ou o Gerenciador de sistemas da ONTAP, pois o Trident não poderá pegar essas alterações.

```
cat backend-san.json
{
    "version": 1,
    "storageDriverName": "ontap-san",
    "backendName": "ontap_san_chap",
```

```
    "managementLIF": "192.168.0.135",
    "svm": "ontap_iscsi_svm",
    "useCHAP": true,
    "username": "vsadmin",
    "password": "password",
    "chapInitiatorSecret": "cl9qxUpDaTeD",
    "chapTargetInitiatorSecret": "rqxigXgkeUpDaTeD",
    "chapTargetUsername": "iJF4heBRT0TCwxyz",
    "chapUsername": "uh2aNCLSd6cNwxyz",
}

./tridentctl update backend ontap_san_chap -f backend-san.json -n trident
+----------------+----------------+--------------------------------------
+--------+---------+
|   NAME         | STORAGE DRIVER |                 UUID                 |
STATE   | VOLUMES |
+----------------+----------------+--------------------------------------
+--------+---------+
| ontap_san_chap | ontap-san      | aa458f3b-ad2d-4378-8a33-1a472ffbeb5c |
online |       7 |
+----------------+----------------+--------------------------------------
+--------+---------+
```

As conexões existentes não serão afetadas. Elas continuarão ativas se as credenciais forem atualizadas pelo Trident no SVM. As novas conexões usam as credenciais atualizadas e as conexões existentes continuam ativas. Desconetar e reconetar PVS antigos resultará em eles usando as credenciais atualizadas.

Exemplos e opções de configuração de SAN ONTAP

Crie e utilize os drivers ONTAP SAN com sua instalação do Trident. Esta seção fornece exemplos de configuração de backend e detalhes para mapear backends para StorageClasses. "Sistemas ASA r2" diferem de outros sistemas ONTAP (ASA, AFF e FAS) na implementação de sua camada de storage. Essas variações impactam o uso de certos parâmetros conforme indicado. "Saiba mais sobre as diferenças entre os sistemas ASA r2 e outros sistemas ONTAP". Na configuração de backend do Trident, você não precisa especificar que seu sistema é ASA r2. Ao selecionar `ontap-san` como o `storageDriverName`, o Trident detecta automaticamente o ASA r2 ou outros sistemas ONTAP. Alguns parâmetros de configuração de backend não se aplicam a sistemas ASA r2, conforme observado na tabela abaixo.

OBSERVAÇÃO	Apenas o `ontap-san` O driver (com protocolos iSCSI, NVMe/TCP e FC) é compatível com sistemas ASA r2.

Opções de configuração de back-end

A partir da versão 26.06, você pode habilitar o posicionamento balanceado do ONTAP para que o ONTAP selecione o agregado para novos volumes. Consulte "Utilize o posicionamento balanceado com os backends

ONTAP" para requisitos e exemplos.

Consulte a tabela a seguir para obter as opções de configuração de back-end:

Parâmetro	Descrição	Padrão
`version`		Sempre 1
`storageDriverName`	Nome do driver de armazenamento. Não pode ser modificado após a criação. Para atualizar este parâmetro, você deve criar um novo backend.	`ontap-san` ou `ontap-san-economy`
`backendName`	Nome personalizado ou back-end de storage	Nome do driver e dataLIF
`managementLIF`	Endereço IP de um cluster ou LIF de gerenciamento de SVM. Um nome de domínio totalmente qualificado (FQDN) pode ser especificado. Pode ser definido para usar endereços IPv6 se o Trident tiver sido instalado usando o sinalizador IPv6. Os endereços IPv6 devem ser definidos entre colchetes, como `[28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]` . Para o switchover MetroCluster otimizado, consulte o Exemplo de MetroCluster. **OBSERVAÇÃO** Se você estiver usando credenciais "vsadmin", `managementLIF` deve ser a do SVM; se estiver usando credenciais "admin", `managementLIF` deve ser a do cluster.	"10,0.0,1", "[2001:1234:abcd::fefe]"
`dataLIF`	Endereço IP do protocolo LIF. Pode ser definido para usar endereços IPv6 se o Trident tiver sido instalado usando o sinalizador IPv6. Os endereços IPv6 devem ser definidos entre colchetes, como `[28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]` . **Não especifique para iSCSI.** O Trident usa "Mapa de LUN seletivo da ONTAP" para descobrir as LIFs iSCSI necessárias para estabelecer uma sessão de vários caminhos. Um aviso é gerado se `dataLIF` for definido explicitamente. **Omita para MetroCluster.** Consulte Exemplo de MetroCluster.	Derivado do SVM
`svm`	Máquina virtual de armazenamento para usar **omit for MetroCluster.** Consulte Exemplo de MetroCluster.	Derivado se uma SVM `managementLIF` for especificada

Parâmetro	Descrição	Padrão
`useCHAP`	Use CHAP para autenticar iSCSI para drivers SAN ONTAP [Boolean]. Defina como `true` para Trident para configurar e usar CHAP bidirecional como a autenticação padrão para o SVM dado no back-end. "Prepare-se para configurar o back-end com drivers SAN ONTAP"Consulte para obter detalhes. **Não compatível com FCP ou NVMe/TCP.**	`false`
`chapInitiatorSecret`	Segredo do iniciador CHAP. Necessário se `useCHAP=true`	""
`labels`	Conjunto de rótulos arbitrários formatados em JSON para aplicar em volumes	""
`chapTargetInitiatorSecret`	Segredo do iniciador de destino CHAP. Necessário se `useCHAP=true`	""
`chapUsername`	Nome de utilizador de entrada. Necessário se `useCHAP=true`	""
`chapTargetUsername`	Nome de utilizador alvo. Necessário se `useCHAP=true`	""
`clientCertificate`	Valor codificado em base64 do certificado do cliente. Usado para autenticação baseada em certificado	""
`clientPrivateKey`	Valor codificado em base64 da chave privada do cliente. Usado para autenticação baseada em certificado	""
`trustedCACertificate`	Valor codificado em base64 do certificado CA confiável. Opcional. Usado para autenticação baseada em certificado.	""
`username`	Nome de usuário necessário para se comunicar com o cluster ONTAP . Usado para autenticação baseada em credenciais. Para autenticação do Active Directory, consulte "Autenticar o Trident em um SVM de backend usando credenciais do Active Directory".	""
`password`	Senha necessária para se comunicar com o cluster ONTAP . Usado para autenticação baseada em credenciais. Para autenticação do Active Directory, consulte "Autenticar o Trident em um SVM de backend usando credenciais do Active Directory".	""
`svm`	Máquina virtual de armazenamento para usar	Derivado se uma SVM `managementLIF` for especificada
`storagePrefix`	Prefixo usado ao provisionar novos volumes no SVM. Não pode ser modificado posteriormente. Para atualizar este parâmetro, você precisa criar um novo backend.	`trident`

Parâmetro	Descrição	Padrão
`aggregate`	Agregado para provisionamento (opcional; se definido, deve ser atribuído à SVM). Para o `ontap-nas-flexgroup` driver, esta opção é ignorada. Se não for atribuído, qualquer um dos agregados disponíveis pode ser usado para provisionar um FlexGroup volume. **OBSERVAÇÃO**: Quando o agregado é atualizado no SVM, ele é atualizado automaticamente no Trident por meio de polling no SVM, sem a necessidade de reiniciar o Trident Controller. Quando você configurou um agregado específico no Trident para provisionar volumes, se o agregado for renomeado ou movido para fora do SVM, o backend entra em estado de falha no Trident durante o polling do agregado no SVM. Você deve alterar o agregado para um que esteja presente no SVM ou removê-lo completamente para que o backend volte a ficar online. **Não especifique para sistemas ASA r2**. **Não especifique quando `useBalancedPlacement` estiver definido como `true`.**	""
`useBalancedPlacement`	Delegar a seleção de aggregate para novos volumes ao ONTAP [Booleano]. Requer ONTAP 9.19.1 ou posterior e `useREST` definido como `true`. Não pode ser combinado com `aggregate`. Consulte "Utilize o posicionamento balanceado com os backends ONTAP" para obter requisitos e exemplos.	`false`

Parâmetro	Descrição	Padrão
`limitAggregateUsage`	Falha no provisionamento se o uso estiver acima dessa porcentagem. Se você estiver usando um back-end do Amazon FSX for NetApp ONTAP, não `limitAggregateUsage`especifique . O fornecido `fsxadmin` e `vsadmin` não contém as permissões necessárias para recuperar o uso agregado e limitá-lo usando o Trident. **Não especifique para sistemas ASA r2**.	"" (não aplicado por padrão)
`limitVolumeSize`	Falha no provisionamento se o tamanho do volume solicitado estiver acima desse valor. Também restringe o tamanho máximo dos volumes que gerencia para LUNs.	"" (não aplicado por padrão)
`lunsPerFlexvol`	Máximo de LUNs por FlexVol, tem de estar no intervalo [50, 200]	`100`
`debugTraceFlags`	Debug flags para usar ao solucionar problemas. Por exemplo, não use a menos que você esteja solucionando problemas e exija um despejo de log detalhado.	`null`

Parâmetro	Descrição	Padrão
`useREST`	Parâmetro booleano para usar APIs REST do ONTAP . `useREST`Quando definido para `true` , O Trident usa APIs REST ONTAP para se comunicar com o backend; quando definido como `false` O Trident usa chamadas ONTAPI (ZAPI) para se comunicar com o backend. Este recurso requer o ONTAP 9.11.1 e posterior. Além disso, a função de login ONTAP usada deve ter acesso ao `ontapi` aplicativo. Isto é satisfeito pelo pré-definido `vsadmin` e `cluster-admin` papéis. A partir da versão Trident 24.06 e ONTAP 9.15.1 ou posterior, `useREST` está definido para `true` por padrão; mudança `useREST` para `false` para usar chamadas ONTAPI (ZAPI). `useREST`é totalmente qualificado para NVMe/TCP. **OBSERVAÇÃO** O NVMe é compatível apenas com APIs REST ONTAP e não com ONTAPI (ZAPI). **Se especificado, sempre definido como `true` para sistemas ASA r2**.	`true` Para ONTAP 9.15,1 ou posterior, caso contrário `false`.
`sanType`	Use para selecionar `iscsi` iSCSI, `nvme` NVMe/TCP ou `fcp` SCSI por Fibre Channel (FC).	`iscsi` se estiver em branco

Parâmetro	Descrição	Padrão
`formatOptions`	Use `formatOptions` para especificar argumentos de linha de comando para o comando `mkfs`, que é aplicado sempre que um volume é formatado. Isso permite que você formate o volume de acordo com suas preferências. Certifique-se de especificar as formatOptions de forma semelhante às opções do comando mkfs, excluindo o caminho do dispositivo. Exemplo: "-E nodiscard" **Suportado para `ontap-san` e `ontap-san-economy` drivers com protocolo iSCSI. Além disso, com suporte para sistemas ASA r2 ao usar protocolos iSCSI e NVMe/TCP.**	
`limitVolumePoolSize`	Tamanho máximo de FlexVol requestable ao usar LUNs no back-end ONTAP-san-econômico.	"" (não aplicado por padrão)
`denyNewVolumePools`	Restringe a `ontap-san-economy` criação de novos volumes do FlexVol para conter LUNs. Somente Flexvols pré-existentes são usados para provisionar novos PVS.	

Recomendações para o uso de formatOptions

A Trident recomenda as seguintes opções para agilizar o processo de formatação:

- **-E nodiscard (ext3, ext4):** Não tente descartar blocos no momento da criação do sistema de arquivos (o descarte inicial de blocos é útil em dispositivos de estado sólido e armazenamento com provisionamento esparso/dinâmico). Esta opção substitui a opção obsoleta "-K" e é aplicável aos sistemas de arquivos ext3 e ext4.
- **-K (xfs):** Não tente descartar blocos no momento do mkfs. Essa opção se aplica ao sistema de arquivos xfs.

Autenticar o Trident em um SVM de backend usando credenciais do Active Directory

Você pode configurar o Trident para autenticar em um SVM de backend usando credenciais do Active Directory (AD). Antes que uma conta do AD possa acessar o SVM, você deve configurar o acesso do controlador de domínio do AD ao cluster ou SVM. Para administração de cluster com uma conta do AD, você deve criar um túnel de domínio. Consulte "Configurar o acesso do controlador de domínio do Active Directory no ONTAP" para mais detalhes.

passos

1. Configurar as definições do Sistema de Nomes de Domínio (DNS) para um SVM de backend:

```
vserver services dns create -vserver <svm_name> -dns-servers
<dns_server_ip1>,<dns_server_ip2>
```

2. Execute o seguinte comando para criar uma conta de computador para o SVM no Active Directory:

```
vserver active-directory create -vserver DataSVM -account-name ADSERVER1
-domain demo.netapp.com
```

3. Use este comando para criar um usuário ou grupo do AD para gerenciar o cluster ou SVM

```
security login create -vserver <svm_name> -user-or-group-name
<ad_user_or_group> -application <application> -authentication-method domain
-role vsadmin
```

4. No arquivo de configuração do backend do Trident , defina o `username` e `password` parâmetros para o nome do usuário ou grupo do AD e senha, respectivamente.

Opções de configuração de back-end para volumes de provisionamento

Você pode controlar o provisionamento padrão usando essas opções na `defaults` seção da configuração. Para obter um exemplo, consulte os exemplos de configuração abaixo.

Parâmetro	Descrição	Padrão
`spaceAllocation`	Alocação de espaço para LUNs	"true" **Se especificado, defina como `true` para sistemas ASA r2**.
`spaceReserve`	Modo de reserva de espaço; "nenhum" (fino) ou "volume" (grosso). **Definido para `none` para sistemas ASA r2**.	"nenhum"
`snapshotPolicy`	Política de instantâneos a utilizar. **Definido para `none` para sistemas ASA r2**.	"nenhum"
`qosPolicy`	Grupo de políticas de QoS a atribuir aos volumes criados. Escolha uma das qosPolicy ou adaptiveQosPolicy por pool de armazenamento/backend. O uso de grupos de política de QoS com Trident requer o ONTAP 9.8 ou posterior. Você deve usar um grupo de políticas de QoS não compartilhado e garantir que o grupo de políticas seja aplicado individualmente a cada componente. Um grupo de políticas de QoS compartilhado impõe o limite máximo da taxa de transferência total de todos os workloads.	""
`adaptiveQosPolicy`	Grupo de políticas de QoS adaptável a atribuir para volumes criados. Escolha uma das qosPolicy ou adaptiveQosPolicy por pool de armazenamento/backend	""
`snapshotReserve`	Porcentagem de volume reservado para snapshots. **Não especifique para sistemas ASA r2**.	"0" se `snapshotPolicy` for "nenhum", caso contrário ""
`splitOnClone`	Divida um clone de seu pai na criação	"falso"
`encryption`	Habilite a NetApp Volume Encryption (NVE) no novo volume; o padrão é `false`. A NVE deve estar licenciada e habilitada no cluster para usar esta opção. Se a NAE estiver habilitada no backend, qualquer volume provisionado no Trident terá a NAE habilitada. Para mais informações, consulte: "Como o Trident funciona com NVE e NAE".	"falso" **Se especificado, defina como `true` para sistemas ASA r2**.

Parâmetro	Descrição	Padrão
`luksEncryption`	Ativar encriptação LUKS. "Usar a configuração de chave unificada do Linux (LUKS)"Consulte a .	"" **Definido para `false` para sistemas ASA r2**.
`tieringPolicy`	Política de hierarquização para usar "nenhum" **Não especifique para sistemas ASA r2**.	
`nameTemplate`	Modelo para criar nomes de volume personalizados.	""

Exemplos de provisionamento de volume

Aqui está um exemplo com padrões definidos:

```
---
version: 1
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: trident_svm
username: admin
password: <password>
labels:
  k8scluster: dev2
  backend: dev2-sanbackend
storagePrefix: alternate-trident
debugTraceFlags:
  api: false
  method: true
defaults:
  spaceReserve: volume
  qosPolicy: standard
  spaceAllocation: 'false'
  snapshotPolicy: default
  snapshotReserve: '10'
```

OBSERVAÇÃO

Para todos os volumes criados usando o `ontap-san` driver, o Trident adiciona 10% de capacidade extra ao FlexVol para acomodar os metadados do LUN. O LUN é provisionado com o tamanho exato que o usuário solicita no PVC. O Trident adiciona 10% ao FlexVol (exibido como tamanho disponível no ONTAP). Os usuários agora obtêm a quantidade de capacidade utilizável que solicitaram. Essa alteração também impede que os LUNs se tornem somente leitura, a menos que o espaço disponível esteja totalmente utilizado. Isso não se aplica ao ontap-san-economy.

Para backends que definem `snapshotReserve`o , o Trident calcula o tamanho dos volumes da seguinte forma:

```
Total volume size = [(PVC requested size) / (1 - (snapshotReserve
```

```
percentage) / 100)] * 1.1
```

O 1.1 representa os 10% extras que a Trident adiciona ao FlexVol para acomodar os metadados do LUN . `snapshotReserve` = 5% e solicitação de PVC = 5 GiB, o tamanho total do volume é 5,79 GiB e o tamanho disponível é 5,5 GiB . `volume show` o comando deve mostrar resultados semelhantes a este exemplo:



Atualmente, o redimensionamento é a única maneira de usar o novo cálculo para um volume existente.

Exemplos mínimos de configuração

Os exemplos a seguir mostram configurações básicas que deixam a maioria dos parâmetros padrão. Esta é a maneira mais fácil de definir um backend.

OBSERVAÇÃO	Se você estiver usando o Amazon FSX no NetApp ONTAP com Trident, o NetApp recomenda que você especifique nomes DNS para LIFs em vez de endereços IP.

▼ **Exemplo de SAN ONTAP**

Esta é uma configuração básica usando `ontap-san` o driver.

```
---
version: 1
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_iscsi
labels:
  k8scluster: test-cluster-1
  backend: testcluster1-sanbackend
username: vsadmin
password: <password>
```

▼ **Exemplo de MetroCluster**

Você pode configurar o back-end para evitar ter que atualizar manualmente a definição do back-end após o switchover e o switchback durante "Replicação e recuperação da SVM"o .

Para comutação e switchback contínuos, especifique o SVM usando `managementLIF` e omite os `svm` parâmetros. Por exemplo:

```
version: 1
storageDriverName: ontap-san
```

```
managementLIF: 192.168.1.66
username: vsadmin
password: password
```

▼ **Exemplo de economia de SAN ONTAP**

```
version: 1
storageDriverName: ontap-san-economy
managementLIF: 10.0.0.1
svm: svm_iscsi_eco
username: vsadmin
password: <password>
```

▼ **Exemplo de autenticação baseada em certificado**

Neste exemplo de configuração básica `clientCertificate` , `clientPrivateKey` e `trustedCACertificate` (opcional, se estiver usando CA confiável) são preenchidos `backend.json` e recebem os valores codificados em base64 do certificado do cliente, da chave privada e do certificado de CA confiável, respetivamente.

```
---
version: 1
storageDriverName: ontap-san
backendName: DefaultSANBackend
managementLIF: 10.0.0.1
svm: svm_iscsi
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSd6cNwxyz
clientCertificate: ZXR0ZXJwYXB...ICMgJ3BhcGVyc2
clientPrivateKey: vciwKIyAgZG...0cnksIGRlc2NyaX
trustedCACertificate: zcyBbaG...b3Igb3duIGNsYXNz
```

▼ **Exemplos CHAP bidirecional**

Esses exemplos criam um backend com `useCHAP` definido como `true`.

Exemplo de ONTAP SAN CHAP

```
---
version: 1
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_iscsi
labels:
```

```
  k8scluster: test-cluster-1
  backend: testcluster1-sanbackend
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSd6cNwxyz
username: vsadmin
password: <password>
```

Exemplo de CHAP de economia de SAN ONTAP

```
---
version: 1
storageDriverName: ontap-san-economy
managementLIF: 10.0.0.1
svm: svm_iscsi_eco
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSd6cNwxyz
username: vsadmin
password: <password>
```

▼ **Exemplo de colocação equilibrada**

Este exemplo habilita o posicionamento balanceado do ONTAP, de forma que o ONTAP selecione o agregado para novos volumes. O posicionamento balanceado requer o ONTAP 9.19.1 ou posterior. Consulte "Utilize o posicionamento balanceado com os backends ONTAP" para obter detalhes.

```
---
version: 1
storageDriverName: ontap-san
backendName: san-balanced-placement
managementLIF: 10.0.0.1
svm: svm_iscsi
username: vsadmin
password: <password>
useBalancedPlacement: true
```

▼ **Exemplo de NVMe/TCP**

Você precisa ter um SVM configurado com NVMe no back-end do ONTAP. Esta é uma configuração básica de back-end para NVMe/TCP.

```
---
version: 1
backendName: NVMeBackend
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_nvme
username: vsadmin
password: password
sanType: nvme
useREST: true
```

▼ **Exemplo de SCSI em FC (FCP)**

Você precisa ter um SVM configurado com FC no back-end do ONTAP. Essa é uma configuração básica de back-end para FC.

```
---
version: 1
backendName: fcp-backend
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_fc
username: vsadmin
password: password
sanType: fcp
useREST: true
```

▼ **Exemplo de configuração de backend com nameTemplate**

```
---
version: 1
storageDriverName: ontap-san
backendName: ontap-san-backend
managementLIF: <ip address>
svm: svm0
username: <admin>
password: <password>
defaults:
  nameTemplate:
"{{.volume.Name}}_{{.labels.cluster}}_{{.volume.Namespace}}_{{.vo\
    lume.RequestName}}"
labels:
  cluster: ClusterA
  PVC: "{{.volume.Namespace}}_{{.volume.RequestName}}"
```

▼ **Exemplo de formatOptions para o driver ONTAP-san-Economy**

```
---
version: 1
storageDriverName: ontap-san-economy
managementLIF: ""
svm: svm1
username: ""
password: "!"
storagePrefix: whelk_
debugTraceFlags:
  method: true
  api: true
defaults:
  formatOptions: -E nodiscard
```

Exemplos de backends com pools virtuais

Nesses arquivos de definição de back-end de exemplo, padrões específicos são definidos para todos os pools de armazenamento, como `spaceReserve` em nenhum, `spaceAllocation` em falso e `encryption` em falso. Os pools virtuais são definidos na seção armazenamento.

O Trident define rótulos de provisionamento no campo "Comentários". Os comentários são definidos nas cópias do FlexVol volume Trident todas as etiquetas presentes em um pool virtual para o volume de storage no provisionamento. Por conveniência, os administradores de storage podem definir rótulos por pool virtual e volumes de grupo por rótulo.

Nesses exemplos, alguns dos pools de armazenamento definem seus próprios `spaceReserve` `spaceAllocation` valores , e `encryption` , e alguns pools substituem os valores padrão.

▼ **Exemplo de SAN ONTAP**

```
---
version: 1
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_iscsi
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSd6cNwxyz
username: vsadmin
password: <password>
defaults:
  spaceAllocation: "false"
  encryption: "false"
  qosPolicy: standard
```

```
labels:
  store: san_store
  kubernetes-cluster: prod-cluster-1
region: us_east_1
storage:
  - labels:
      protection: gold
      creditpoints: "40000"
    zone: us_east_1a
    defaults:
      spaceAllocation: "true"
      encryption: "true"
      adaptiveQosPolicy: adaptive-extreme
  - labels:
      protection: silver
      creditpoints: "20000"
    zone: us_east_1b
    defaults:
      spaceAllocation: "false"
      encryption: "true"
      qosPolicy: premium
  - labels:
      protection: bronze
      creditpoints: "5000"
    zone: us_east_1c
    defaults:
      spaceAllocation: "true"
      encryption: "false"
```

▼ **Exemplo de economia de SAN ONTAP**

```
---
version: 1
storageDriverName: ontap-san-economy
managementLIF: 10.0.0.1
svm: svm_iscsi_eco
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSd6cNwxyz
username: vsadmin
password: <password>
defaults:
  spaceAllocation: "false"
  encryption: "false"
```

```
labels:
  store: san_economy_store
region: us_east_1
storage:
  - labels:
      app: oracledb
      cost: "30"
    zone: us_east_1a
    defaults:
      spaceAllocation: "true"
      encryption: "true"
  - labels:
      app: postgresdb
      cost: "20"
    zone: us_east_1b
    defaults:
      spaceAllocation: "false"
      encryption: "true"
  - labels:
      app: mysqldb
      cost: "10"
    zone: us_east_1c
    defaults:
      spaceAllocation: "true"
      encryption: "false"
  - labels:
      department: legal
      creditpoints: "5000"
    zone: us_east_1c
    defaults:
      spaceAllocation: "true"
      encryption: "false"
```

▼ **Exemplo de NVMe/TCP**

```
---
version: 1
storageDriverName: ontap-san
sanType: nvme
managementLIF: 10.0.0.1
svm: nvme_svm
username: vsadmin
password: <password>
useREST: true
defaults:
  spaceAllocation: "false"
```

```
  encryption: "true"
storage:
  - labels:
      app: testApp
      cost: "20"
    defaults:
      spaceAllocation: "false"
      encryption: "false"
```

Mapeie os backends para StorageClasses

As seguintes definições de StorageClass referem-se ao Exemplos de backends com pools virtuais. Usando o campo `parameters.selector`, cada StorageClass indica quais pools virtuais podem ser usados para hospedar um volume. O volume agora possui os aspectos definidos no pool virtual escolhido.

- O `protection-gold` StorageClass corresponde ao primeiro pool virtual no `ontap-san` backend. Este é o único pool que oferece proteção de nível ouro.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-gold
provisioner: csi.trident.netapp.io
parameters:
  selector: "protection=gold"
  fsType: "ext4"
```

- O `protection-not-gold` StorageClass corresponde ao segundo e terceiro pool virtual no `ontap-san` backend. Estes são os únicos pools que oferecem um nível de proteção diferente do gold.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-not-gold
provisioner: csi.trident.netapp.io
parameters:
  selector: "protection!=gold"
  fsType: "ext4"
```

- O `app-mysqldb` StorageClass corresponde ao terceiro pool virtual no `ontap-san-economy` backend. Este é o único pool que oferece configuração de pool de storage para o aplicativo do tipo mysqldb.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
```

```
  name: app-mysqldb
provisioner: csi.trident.netapp.io
parameters:
  selector: "app=mysqldb"
  fsType: "ext4"
```

- O `protection-silver-creditpoints-20k` StorageClass corresponde ao segundo pool virtual no `ontap-san` backend. Este é o único pool que oferece proteção de nível prata e 20000 pontos de crédito.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-silver-creditpoints-20k
provisioner: csi.trident.netapp.io
parameters:
  selector: "protection=silver; creditpoints=20000"
  fsType: "ext4"
```

- O `creditpoints-5k` StorageClass corresponde ao terceiro pool virtual no `ontap-san` backend e ao quarto pool virtual no `ontap-san-economy` backend. Essas são as únicas ofertas de pool com 5000 creditpoints.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: creditpoints-5k
provisioner: csi.trident.netapp.io
parameters:
  selector: "creditpoints=5000"
  fsType: "ext4"
```

- O `my-test-app-sc` StorageClass corresponde ao `testAPP` pool virtual no `ontap-san` driver com `sanType: nvme`. Este é o único pool que oferece `testApp`.

```
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: my-test-app-sc
provisioner: csi.trident.netapp.io
parameters:
  selector: "app=testApp"
  fsType: "ext4"
```

O Trident decide qual pool virtual será selecionado e garante que o requisito de storage seja atendido.

Atualizar LIFs de dados NVMe

Habilite a atualização de LIFs de dados NVMe para permitir que volumes NVMe publicados usem LIFs que foram adicionadas, removidas ou que receberam um novo endereço sem recriar o backend ou reiniciar o Trident.

Visão geral

Ao adicionar um backend SAN do ONTAP, o Trident descobre as LIFs de dados NVMe para a SVM. Por padrão, o Trident retém esse conjunto de LIFs até que você adicione o backend novamente ou reinicie o Trident.

Defina `enableDataLIFRefresh` como `true` para atualizar as LIFs de dados NVMe enquanto o Trident estiver em execução. Os volumes NVMe publicados usam então as LIFs que são adicionadas, removidas ou às quais é atribuído um novo endereço.

Essa opção é um parâmetro de instalação, não um parâmetro de configuração de backend. Ela está desativada por padrão.

A atualização da LIF de dados NVMe está disponível a partir do Trident 26.10.

Esta funcionalidade aplica-se a:

- `ontap-san` com `sanType: nvme` no ONTAP unificado.
- `ontap-san` usando NVMe/TCP no ASA r2.

CUIDADO	Remover ou substituir todas as LIFs de dados ao mesmo tempo causa interrupção. Se você planeja remover ou substituir todas as LIFs de dados, certifique-se de que uma LIF de substituição esteja em vigor primeiro para que o acesso aos dados não seja interrompido.

Ativar atualização da LIF de dados NVMe

Operador Trident

Para uma nova instalação, defina `enableDataLIFRefresh` na `TridentOrchestrator` especificação:

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  namespace: trident
  enableDataLIFRefresh: true
```

Aplique o manifesto:

```
kubectl apply -f tridentorchestrator_cr.yaml
```

Para uma instalação existente, aplique o patch em `TridentOrchestrator`:

```
kubectl patch torc trident --type=merge -p
'{"spec":{"enableDataLIFRefresh":true}}'
```

Verifique o valor aplicado:

```
kubectl get torc trident -o
jsonpath='{.status.currentInstallationParams.enableDataLIFRefresh}'
```

Leme

Para uma nova instalação, defina o valor do Helm:

```
helm install trident netapp-trident/trident-operator \
  --namespace trident --create-namespace \
  --set enableDataLIFRefresh=true
```

Para uma instalação existente do Helm, atualize a release com o valor habilitado:

```
helm upgrade trident netapp-trident/trident-operator \
  --namespace trident \
  --reuse-values \
  --set enableDataLIFRefresh=true
```

Você também pode definir `enableDataLIFRefresh: true` em um arquivo de valores do Helm e passá-lo para `helm upgrade` com `-f <values-file>`.

tridentctl

Para uma nova instalação, use a `--enable-datalif-refresh` flag:

```
tridentctl install --enable-datalif-refresh -n trident
```

Para habilitar a opção em uma instalação `tridentctl` existente, desinstale o Trident e instale-o novamente com a flag. Siga as precauções em "Desinstale o Trident" antes de desinstalar.

```
tridentctl uninstall -n trident
```

```
tridentctl install --enable-datalif-refresh -n trident
```

OBSERVAÇÃO A `--enable-datalif-refresh`flag é uma opção de instalação e não é exibida em `tridentctl install --help`.

Drivers nas ONTAP

Descrição geral do controlador ONTAP nas

Saiba mais sobre como configurar um back-end ONTAP com drivers nas ONTAP e Cloud Volumes ONTAP.

Detalhes do driver nas do ONTAP

O Trident fornece os seguintes drivers de armazenamento nas para se comunicar com o cluster ONTAP. Os modos de acesso suportados são: *ReadWriteOnce* (RWO), *ReadOnlyMuy* (ROX), *ReadWriteMany* (RWX), *ReadWriteOncePod* (RWOP).

Condutor	Protocolo	VolumeMode	Modos de acesso suportados	Sistemas de arquivos suportados
`ontap-nas`	NFS, SMB	Sistema de ficheiros	RWO, ROX, RWX, RWOP	"" `nfs`, , `smb`
`ontap-nas-economy`	NFS, SMB	Sistema de ficheiros	RWO, ROX, RWX, RWOP	"" `nfs`, , `smb`
`ontap-nas-flexgroup`	NFS, SMB	Sistema de ficheiros	RWO, ROX, RWX, RWOP	"" `nfs`, , `smb`

AVISO

- Use `ontap-san-economy` somente se a contagem de uso de volume persistente for esperada ser maior que "Limites de volume ONTAP suportados".
- Use `ontap-nas-economy` somente se a contagem de uso de volume persistente for esperada para ser maior do que "Limites de volume ONTAP suportados" e o `ontap-san-economy` driver não puder ser usado.
- Não use o uso `ontap-nas-economy` se você antecipar a necessidade de proteção de dados, recuperação de desastres ou mobilidade.
- O dimensionamento automático de volume do ONTAP não é compatível com os drivers NAS do ONTAP. Para obter informações sobre os drivers SAN compatíveis e detalhes de configuração, consulte "Configurar o dimensionamento automático de volume do ONTAP".

Permissões do usuário

O Trident espera ser executado como administrador do ONTAP ou SVM, normalmente usando o `admin` usuário do cluster ou um `vsadmin` usuário SVM, ou um usuário com um nome diferente que tenha a mesma função.

Para implantações do Amazon FSX for NetApp ONTAP, o Trident espera ser executado como administrador do ONTAP ou SVM, usando o usuário do cluster `fsxadmin` ou um `vsadmin` usuário SVM, ou um usuário com um nome diferente que tenha a mesma função. O `fsxadmin` usuário é um substituto limitado para o usuário

administrador do cluster.

OBSERVAÇÃO	Se você usar o `limitAggregateUsage` parâmetro, as permissões de administrador do cluster serão necessárias. Ao usar o Amazon FSX for NetApp ONTAP com Trident, o `limitAggregateUsage` parâmetro não funcionará com as `vsadmin` contas de usuário e `fsxadmin`. A operação de configuração falhará se você especificar este parâmetro.

Embora seja possível criar uma função mais restritiva no ONTAP que um driver Trident pode usar, não recomendamos. A maioria das novas versões do Trident chamarão APIs adicionais que teriam que ser contabilizadas, tornando as atualizações difíceis e suscetíveis a erros.

Prepare-se para configurar um back-end com drivers nas ONTAP

Compreenda os requisitos, as opções de autenticação e as políticas de exportação para configurar um backend ONTAP com drivers ONTAP NAS. A partir da versão 25.10, NetApp Trident oferece suporte "Sistema de armazenamento NetApp AFX". NetApp AFX storage systems diferem de outros sistemas ONTAP (ASA, AFF e FAS) na implementação de sua camada de storage. Na configuração do backend do Trident, não é necessário especificar que seu sistema é AFX. Ao selecionar `ontap-nas` como o `storageDriverName`, o Trident detecta automaticamente os sistemas AFX.

OBSERVAÇÃO	Apenas o `ontap-nas` O driver (com protocolo NFS) é compatível com sistemas AFX; o protocolo SMB não é compatível.

Requisitos

- Para todos os backends ONTAP, o Trident exige que pelo menos um agregado seja atribuído ao SVM.
- Você pode executar mais de um driver e criar classes de armazenamento que apontam para um ou outro. Por exemplo, você pode configurar uma classe Gold que usa o `ontap-nas` driver e uma classe Bronze que usa o `ontap-nas-economy` um.
- Todos os seus nós de trabalho do Kubernetes precisam ter as ferramentas NFS apropriadas instaladas. "aqui"Consulte para obter mais detalhes.
- O Trident dá suporte a volumes SMB montados em pods executados apenas em nós do Windows. Prepare-se para provisionar volumes SMBConsulte para obter detalhes.

Autenticar o back-end do ONTAP

O Trident oferece dois modos de autenticar um back-end do ONTAP.

- Baseado em credenciais: Esse modo requer permissões suficientes para o back-end do ONTAP. Recomenda-se usar uma conta associada a uma função de login de segurança predefinida, como `admin` ou `vsadmin` para garantir a máxima compatibilidade com as versões do ONTAP.
- Baseado em certificado: Este modo requer um certificado instalado no back-end para que o Trident se comunique com um cluster ONTAP. Aqui, a definição de back-end deve conter valores codificados em Base64 do certificado de cliente, chave e certificado de CA confiável, se usado (recomendado).

Você pode atualizar os backends existentes para mover entre métodos baseados em credenciais e baseados em certificado. No entanto, apenas um método de autenticação é suportado por vez. Para alternar para um

método de autenticação diferente, você deve remover o método existente da configuração de back-end.

AVISO	Se você tentar fornecer **credenciais e certificados**, a criação de back-end falhará com um erro que mais de um método de autenticação foi fornecido no arquivo de configuração.

Ative a autenticação baseada em credenciais

O Trident requer as credenciais para um administrador com escopo SVM/escopo de cluster para se comunicar com o back-end do ONTAP. Recomenda-se a utilização de funções padrão predefinidas, como `admin` ou `vsadmin`. Isso garante compatibilidade direta com futuras versões do ONTAP que podem expor APIs de recursos a serem usadas por futuras versões do Trident. Uma função de login de segurança personalizada pode ser criada e usada com o Trident, mas não é recomendada.

Uma definição de backend de exemplo será assim:

YAML

```yaml
---
version: 1
backendName: ExampleBackend
storageDriverName: ontap-nas
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
svm: svm_nfs
credentials:
  name: secret-backend-creds
```

JSON

```json
{
  "version": 1,
  "backendName": "ExampleBackend",
  "storageDriverName": "ontap-nas",
  "managementLIF": "10.0.0.1",
  "dataLIF": "10.0.0.2",
  "svm": "svm_nfs",
  "credentials": {
        "name": "secret-backend-creds"
    }
}
```

Tenha em mente que a definição de back-end é o único lugar onde as credenciais são armazenadas em texto simples. Depois que o back-end é criado, os nomes de usuário/senhas são codificados com Base64 e armazenados como segredos do Kubernetes. A criação/updation de um backend é a única etapa que requer conhecimento das credenciais. Como tal, é uma operação somente de administrador, a ser realizada pelo administrador do Kubernetes/storage.

Ativar autenticação baseada em certificado

Backends novos e existentes podem usar um certificado e se comunicar com o back-end do ONTAP. Três parâmetros são necessários na definição de backend.

- ClientCertificate: Valor codificado base64 do certificado do cliente.
- ClientPrivateKey: Valor codificado em base64 da chave privada associada.
- TrustedCACertificate: Valor codificado base64 do certificado CA confiável. Se estiver usando uma CA confiável, esse parâmetro deve ser fornecido. Isso pode ser ignorado se nenhuma CA confiável for usada.

Um fluxo de trabalho típico envolve as etapas a seguir.

Passos

1. Gerar um certificado e chave de cliente. Ao gerar, defina Nome Comum (CN) para o usuário ONTAP para autenticar como.

```
openssl req -x509 -nodes -days 1095 -newkey rsa:2048 -keyout k8senv.key
-out k8senv.pem -subj "/C=US/ST=NC/L=RTP/O=NetApp/CN=vsadmin"
```

2. Adicionar certificado de CA confiável ao cluster do ONTAP. Isso pode já ser Tratado pelo administrador do armazenamento. Ignore se nenhuma CA confiável for usada.

```
security certificate install -type server -cert-name <trusted-ca-cert-
name> -vserver <vserver-name>
ssl modify -vserver <vserver-name> -server-enabled true -client-enabled
true -common-name <common-name> -serial <SN-from-trusted-CA-cert> -ca
<cert-authority>
```

3. Instale o certificado e a chave do cliente (a partir do passo 1) no cluster do ONTAP.

```
security certificate install -type client-ca -cert-name <certificate-
name> -vserver <vserver-name>
security ssl modify -vserver <vserver-name> -client-enabled true
```

4. Confirme se a função de login de segurança do ONTAP suporta `cert` o método de autenticação.

```
security login create -user-or-group-name vsadmin -application ontapi
-authentication-method cert -vserver <vserver-name>
security login create -user-or-group-name vsadmin -application http
-authentication-method cert -vserver <vserver-name>
```

5. Teste a autenticação usando certificado gerado. Substitua o ONTAP Management LIF> e o <vserver name> por IP de LIF de gerenciamento e nome da SVM. Você deve garantir que o LIF tenha sua política de serviço definida como `default-data-management`.

```
curl -X POST -Lk https://<ONTAP-Management-
LIF>/servlets/netapp.servlets.admin.XMLrequest_filer --key k8senv.key
--cert ~/k8senv.pem -d '<?xml version="1.0" encoding="UTF-8"?><netapp
xmlns="http://www.netapp.com/filer/admin" version="1.21"
vfiler="<vserver-name>"><vserver-get></vserver-get></netapp>'
```

6. Codificar certificado, chave e certificado CA confiável com Base64.

```
base64 -w 0 k8senv.pem >> cert_base64
base64 -w 0 k8senv.key >> key_base64
base64 -w 0 trustedca.pem >> trustedca_base64
```

7. Crie backend usando os valores obtidos na etapa anterior.

```
cat cert-backend-updated.json
{
"version": 1,
"storageDriverName": "ontap-nas",
"backendName": "NasBackend",
"managementLIF": "1.2.3.4",
"dataLIF": "1.2.3.8",
"svm": "vserver_test",
"clientCertificate": "Faaaakkkkeeee...Vaaalllluuuueeee",
"clientPrivateKey": "LS0tFaKE...0VaLuES0tLS0K",
"storagePrefix": "myPrefix_"
}

#Update backend with tridentctl
tridentctl update backend NasBackend -f cert-backend-updated.json -n
trident
+------------+----------------+--------------------------------------
+--------+---------+
|    NAME    | STORAGE DRIVER |                 UUID                 |
STATE  | VOLUMES |
+------------+----------------+--------------------------------------
+--------+---------+
| NasBackend | ontap-nas      | 98e19b74-aec7-4a3d-8dcf-128e5033b214 |
online |       9 |
+------------+----------------+--------------------------------------
+--------+---------+
```

Atualizar métodos de autenticação ou girar credenciais

Você pode atualizar um back-end existente para usar um método de autenticação diferente ou para girar suas credenciais. Isso funciona de ambas as maneiras: Backends que fazem uso de nome de usuário / senha podem ser atualizados para usar certificados; backends que utilizam certificados podem ser atualizados para nome de usuário / senha com base. Para fazer isso, você deve remover o método de autenticação existente e adicionar o novo método de autenticação. Em seguida, use o arquivo backend.json atualizado contendo os parâmetros necessários para executar `tridentctl update backend`.

```
cat cert-backend-updated.json
```

```
{
"version": 1,
"storageDriverName": "ontap-nas",
"backendName": "NasBackend",
"managementLIF": "1.2.3.4",
"dataLIF": "1.2.3.8",
"svm": "vserver_test",
"username": "vsadmin",
"password": "password",
"storagePrefix": "myPrefix_"
}
```

```
#Update backend with tridentctl
tridentctl update backend NasBackend -f cert-backend-updated.json -n
trident
+------------+----------------+--------------------------------------
+--------+---------+
|    NAME    | STORAGE DRIVER |                 UUID                 |
STATE   | VOLUMES |
+------------+----------------+--------------------------------------
+--------+---------+
| NasBackend | ontap-nas      | 98e19b74-aec7-4a3d-8dcf-128e5033b214 |
online |       9 |
+------------+----------------+--------------------------------------
+--------+---------+
```

OBSERVAÇÃO

Ao girar senhas, o administrador de armazenamento deve primeiro atualizar a senha do usuário no ONTAP. Isso é seguido por uma atualização de back-end. Ao girar certificados, vários certificados podem ser adicionados ao usuário. O back-end é então atualizado para usar o novo certificado, seguindo o qual o certificado antigo pode ser excluído do cluster do ONTAP.

A atualização de um back-end não interrompe o acesso a volumes que já foram criados, nem afeta as conexões de volume feitas depois. Uma atualização de back-end bem-sucedida indica que o Trident pode se

comunicar com o back-end do ONTAP e lidar com operações de volume futuras.

Crie uma função ONTAP personalizada para o Trident

Você pode criar uma função de cluster do ONTAP com Privileges mínimo para que você não precise usar a função de administrador do ONTAP para executar operações no Trident. Quando você inclui o nome de usuário em uma configuração de back-end do Trident, o Trident usa a função de cluster do ONTAP criada para executar as operações.

"Gerador de função personalizada Trident"Consulte para obter mais informações sobre como criar funções personalizadas do Trident.

Usando a CLI do ONTAP

1. Crie uma nova função usando o seguinte comando:

```
security login role create <role_name\> -cmddirname "command" -access all
–vserver <svm_name\>
```

2. Crie um nome de usuário para o usuário do Trident:

```
security login create -username <user_name\> -application ontapi
-authmethod <password\> -role <name_of_role_in_step_1\> –vserver
<svm_name\> -comment "user_description"
```

3. Mapeie a função para o usuário:

```
security login modify username <user_name\> –vserver <svm_name\> -role
<role_name\> -application ontapi -application console -authmethod
<password\>
```

Usando o System Manager

Execute as seguintes etapas no Gerenciador do sistema do ONTAP:

1. **Crie uma função personalizada**:
   a. Para criar uma função personalizada no nível do cluster, selecione **Cluster > Settings**.

      (Ou) para criar uma função personalizada no nível SVM, selecione **Storage > Storage VMs > `required SVM` Settings > Users and Roles**.

   b. Selecione o ícone de seta (→) ao lado de **usuários e funções**.
   c. Selecione * Adicionar * em **funções**.
   d. Defina as regras para a função e clique em **Salvar**.
2. **Mapeie a função para o usuário do Trident**: Execute as seguintes etapas na página **usuários e funções**:
   a. Selecione Adicionar ícone *** em *usuários**.
   b. Selecione o nome de usuário desejado e selecione uma função no menu suspenso para **função**.
   c. Clique em **Salvar**.

Consulte as páginas a seguir para obter mais informações:

- "Funções personalizadas para administração do ONTAP" ou "Definir funções personalizadas"
- "Trabalhe com funções e usuários"

Gerenciar políticas de exportação de NFS

O Trident usa políticas de exportação de NFS para controlar o acesso aos volumes provisionados.

O Trident fornece duas opções ao trabalhar com políticas de exportação:

- O Trident pode gerenciar dinamicamente a própria política de exportação; nesse modo de operação, o administrador de armazenamento especifica uma lista de blocos CIDR que representam endereços IP admissíveis. O Trident adiciona IPs de nós aplicáveis que se enquadram nesses intervalos à política de exportação automaticamente no momento da publicação. Como alternativa, quando nenhum CIDR é especificado, todos os IPs unicast de escopo global encontrados no nó para o qual o volume será publicado serão adicionados à política de exportação.
- Os administradores de storage podem criar uma política de exportação e adicionar regras manualmente. O Trident usa a política de exportação padrão, a menos que um nome de política de exportação diferente seja especificado na configuração.

Gerencie dinamicamente políticas de exportação

O Trident fornece a capacidade de gerenciar dinamicamente políticas de exportação para backends ONTAP. Isso fornece ao administrador de armazenamento a capacidade de especificar um espaço de endereço permitido para IPs de nó de trabalho, em vez de definir regras explícitas manualmente. Ele simplifica muito o gerenciamento de políticas de exportação. As modificações na política de exportação não exigem mais intervenção manual no cluster de storage. Além disso, isso ajuda a restringir o acesso ao cluster de armazenamento somente aos nós de trabalho que estão montando volumes e têm IPs no intervalo especificado, suportando um gerenciamento refinado e automatizado.

OBSERVAÇÃO	Não use NAT (Network Address Translation) ao usar políticas de exportação dinâmicas. Com o NAT, o controlador de armazenamento vê o endereço NAT frontend e não o endereço IP real do host, portanto, o acesso será negado quando nenhuma correspondência for encontrada nas regras de exportação.

Exemplo

Há duas opções de configuração que devem ser usadas. Aqui está um exemplo de definição de backend:

```
---
version: 1
storageDriverName: ontap-nas-economy
backendName: ontap_nas_auto_export
managementLIF: 192.168.0.135
svm: svm1
username: vsadmin
password: password
autoExportCIDRs:
  - 192.168.0.0/24
autoExportPolicy: true
```

OBSERVAÇÃO	Ao usar esse recurso, você deve garantir que a junção raiz do SVM tenha uma política de exportação criada anteriormente com uma regra de exportação que permita o bloco CIDR do nó (como a política de exportação padrão). Siga sempre as melhores práticas recomendadas pela NetApp para dedicar um SVM para Trident.

Aqui está uma explicação de como esse recurso funciona usando o exemplo acima:

- `autoExportPolicy` está definido como `true`. Isso indica que o Trident cria uma política de exportação para cada volume provisionado com esse back-end para `svm1` o SVM e lida com a adição e exclusão de regras usando `autoexportCIDRs` blocos de endereço. Até que um volume seja anexado a um nó, o volume usa uma política de exportação vazia sem regras para impedir o acesso indesejado a esse volume. Quando um volume é publicado em um nó, o Trident cria uma política de exportação com o mesmo nome que a qtree subjacente que contém o IP do nó dentro do bloco CIDR especificado. Esses IPs também serão adicionados à política de exportação usada pelo FlexVol volume pai
  - Por exemplo:
    - Back-end UUID 403b5326-8482-40dB-96d0-d83fb3f4daec
    - `autoExportPolicy` defina como `true`
    - prefixo de armazenamento `trident`
    - PVC UUID a79bcf5f-7b6d-4a40-9876-e2551f159c1c
    - A qtree Trident_pvc_a79bcf5f_7b6d_4a40_9876_e2551f159c1c cria uma política de exportação para o FlexVol `trident-403b5326-8482-40db96d0-d83fb3f4daec` nomeado , uma política de exportação para a qtree `trident_pvc_a79bcf5f_7b6d_4a40_9876_e2551f159c1c` nomeada e uma política de exportação vazia nomeada `trident_empty` na SVM. As regras para a política de exportação do FlexVol serão um superconjunto de quaisquer regras contidas nas políticas de exportação de qtree. A política de exportação vazia será reutilizada por quaisquer volumes que não estejam anexados.
- `autoExportCIDRs` contém uma lista de blocos de endereços. Este campo é opcional e o padrão é ["0,0.0,0/0", "::/0"]. Se não estiver definido, o Trident adiciona todos os endereços unicast de escopo global encontrados nos nós de trabalho com publicações.

Neste exemplo, o `192.168.0.0/24` espaço de endereço é fornecido. Isso indica que os IPs de nó do Kubernetes que se enquadram nesse intervalo de endereços com publicações serão adicionados à política de exportação criada pelo Trident. Quando o Trident Registra um nó em que ele é executado, ele recupera os endereços IP do nó e os verifica em relação aos blocos de endereços fornecidos no `autoExportCIDRs`. no momento da publicação, após filtrar os IPs, o Trident cria as regras de política de exportação para os IPs do cliente para o nó em que está publicando.

Você pode atualizar `autoExportPolicy` e `autoExportCIDRs` para backends depois de criá-los. Você pode anexar novos CIDR para um back-end que é gerenciado automaticamente ou excluir CIDR existentes. Tenha cuidado ao excluir CIDR para garantir que as conexões existentes não sejam descartadas. Você também pode optar por desativar `autoExportPolicy` um back-end e retornar a uma política de exportação criada manualmente. Isso exigirá a configuração do `exportPolicy` parâmetro em sua configuração de backend.

Depois que o Trident cria ou atualiza um backend, você pode verificar o backend usando `tridentctl` ou o CRD correspondente `tridentbackend`:

```
./tridentctl get backends ontap_nas_auto_export -n trident -o yaml
```

```
items:
- backendUUID: 403b5326-8482-40db-96d0-d83fb3f4daec
  config:
    aggregate: ""
    autoExportCIDRs:
    - 192.168.0.0/24
    autoExportPolicy: true
    backendName: ontap_nas_auto_export
    chapInitiatorSecret: ""
    chapTargetInitiatorSecret: ""
    chapTargetUsername: ""
    chapUsername: ""
    dataLIF: 192.168.0.135
    debug: false
    debugTraceFlags: null
    defaults:
      encryption: "false"
      exportPolicy: <automatic>
      fileSystemType: ext4
```

Quando um nó é removido, o Trident verifica todas as políticas de exportação para remover as regras de acesso correspondentes ao nó. Ao remover esse IP de nó das políticas de exportação de backends gerenciados, o Trident impede montagens fraudulentas, a menos que esse IP seja reutilizado por um novo nó no cluster.

Para backends existentes anteriormente, atualizar o backend com `tridentctl update backend` garante que o Trident gerencia as políticas de exportação automaticamente. Isso cria duas novas políticas de exportação nomeadas após o UUID e o nome de qtree do back-end quando elas são necessárias. Os volumes presentes no back-end usarão as políticas de exportação recém-criadas depois que forem desmontadas e montadas novamente.

OBSERVAÇÃO	A exclusão de um back-end com políticas de exportação gerenciadas automaticamente excluirá a política de exportação criada dinamicamente. Se o backend for recriado, ele será Tratado como um novo backend e resultará na criação de uma nova política de exportação.

Se o endereço IP de um nó ativo for atualizado, você deverá reiniciar o pod Trident no nó. O Trident atualizará então a política de exportação para backends que consegue refletir esta alteração de IP.

Prepare-se para provisionar volumes SMB

Com um pouco de preparação adicional, você pode provisionar volumes SMB usando `ontap-nas` drivers.

AVISO	É necessário configurar os protocolos NFS e SMB/CIFS na SVM para criar um `ontap-nas-economy` volume SMB para clusters no local do ONTAP. A falha na configuração desses protocolos fará com que a criação de volume SMB falhe.

OBSERVAÇÃO	`autoExportPolicy` Não é compatível com volumes SMB.

Antes de começar

Antes de provisionar volumes SMB, você deve ter o seguinte:

- Um cluster do Kubernetes com um nó de controlador Linux e pelo menos um nó de trabalho do Windows que executa o Windows Server 2022. O Trident dá suporte a volumes SMB montados em pods executados apenas em nós do Windows.
- Pelo menos um segredo do Trident contendo suas credenciais do ative Directory. Para gerar segredo `smbcreds`:

```
kubectl create secret generic smbcreds --from-literal username=user
--from-literal password='password'
```

- Um proxy CSI configurado como um serviço Windows. Para configurar um `csi-proxy`, "GitHub: CSI Proxy"consulte ou "GitHub: CSI Proxy para Windows" para nós do Kubernetes executados no Windows.

Passos

1. Para o ONTAP no local, você pode criar, opcionalmente, um compartilhamento SMB ou o Trident pode criar um para você.

   OBSERVAÇÃO Compartilhamentos SMB são necessários para o Amazon FSX for ONTAP.

   Você pode criar os compartilhamentos de administração SMB de duas maneiras usando o "Microsoft Management Console"snap-in pastas compartilhadas ou usando a CLI do ONTAP. Para criar compartilhamentos SMB usando a CLI do ONTAP:

   a. Se necessário, crie a estrutura do caminho do diretório para o compartilhamento.

      O `vserver cifs share create` comando verifica o caminho especificado na opção -path durante a criação de compartilhamento. Se o caminho especificado não existir, o comando falhará.

   b. Crie um compartilhamento SMB associado ao SVM especificado:

```
vserver cifs share create -vserver vserver_name -share-name
share_name -path path [-share-properties share_properties,...]
[other_attributes] [-comment text]
```

   c. Verifique se o compartilhamento foi criado:

```
vserver cifs share show -share-name share_name
```

   OBSERVAÇÃO "Crie um compartilhamento SMB"Consulte para obter detalhes completos.

2. Ao criar o back-end, você deve configurar o seguinte para especificar volumes SMB. Para obter todas as opções de configuração de back-end do FSX for ONTAP, "Opções e exemplos de configuração do FSX for ONTAP"consulte .

Parâmetro	Descrição	Exemplo
`smbShare`	Você pode especificar uma das seguintes opções: O nome de um compartilhamento SMB criado usando o Console de Gerenciamento da Microsoft ou a CLI do ONTAP; um nome para permitir que o Trident crie o compartilhamento SMB; ou você pode deixar o parâmetro em branco para impedir o acesso comum ao compartilhamento a volumes. Esse parâmetro é opcional para o ONTAP no local. Esse parâmetro é necessário para backends do Amazon FSX for ONTAP e não pode ficar em branco.	`smb-share`
`nasType`	**Tem de estar definido para `smb`.** Se nulo, o padrão é `nfs`.	`smb`
`securityStyle`	Estilo de segurança para novos volumes. **Deve ser definido como `ntfs` ou `mixed` para volumes SMB.**	`ntfs` Ou `mixed` para volumes SMB
`unixPermissions`	Modo para novos volumes. **Deve ser deixado vazio para volumes SMB.**	""

Habilitar SMB seguro

A partir da versão 25.06, o NetApp Trident oferece suporte ao provisionamento seguro de volumes SMB criados usando `ontap-nas` e `ontap-nas-economy` backends. Quando o SMB seguro está habilitado, você pode fornecer acesso controlado aos compartilhamentos SMB para usuários e grupos de usuários do Active Directory (AD) usando Listas de Controle de Acesso (ACLs).

Pontos a lembrar

- Importando `ontap-nas-economy` volumes não são suportados.
- Somente clones somente leitura são suportados para `ontap-nas-economy` volumes.
- Se o Secure SMB estiver habilitado, o Trident ignorará o compartilhamento SMB mencionado no backend.
- Atualizar a anotação de PVC, a anotação de classe de armazenamento e o campo de backend não atualiza a ACL de compartilhamento SMB.
- A ACL de compartilhamento SMB especificada na anotação do PVC clone terá precedência sobre aquelas no PVC de origem.
- Certifique-se de fornecer usuários válidos do AD ao habilitar o SMB seguro. Usuários inválidos não serão adicionados à ACL.
- Se você fornecer ao mesmo usuário do AD no backend, classe de armazenamento e PVC permissões diferentes, a prioridade de permissão será: PVC, classe de armazenamento e, em seguida, backend.
- O Secure SMB é suportado para `ontap-nas` importações de volume gerenciado e não aplicável a importações de volume não gerenciado.

Passos

1. Especifique adAdminUser em TridentBackendConfig, conforme mostrado no exemplo a seguir:

```
apiVersion: trident.netapp.io/v1
```

```
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-nas
  managementLIF: 10.193.176.x
  svm: svm0
  useREST: true
  defaults:
    adAdminUser: tridentADtest
  credentials:
    name: backend-tbc-ontap-invest-secret
```

2. Adicione a anotação na classe de armazenamento.

   Adicione o `trident.netapp.io/smbShareAdUser` anotação para a classe de armazenamento para habilitar o SMB seguro sem falhas. O valor do usuário especificado para a anotação `trident.netapp.io/smbShareAdUser` deve ser o mesmo que o nome de usuário especificado no `smbcreds` segredo. é `full_control` .

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-smb-sc
  annotations:
    trident.netapp.io/smbShareAdUserPermission: change
    trident.netapp.io/smbShareAdUser: tridentADuser
parameters:
  backendType: ontap-nas
  csi.storage.k8s.io/node-stage-secret-name: smbcreds
  csi.storage.k8s.io/node-stage-secret-namespace: trident
  trident.netapp.io/nasType: smb
provisioner: csi.trident.netapp.io
reclaimPolicy: Delete
volumeBindingMode: Immediate
```

1. Crie um PVC.

   O exemplo a seguir cria um PVC:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
```

```
  name: my-pvc4
  namespace: trident
  annotations:
    trident.netapp.io/snapshotDirectory: "true"
    trident.netapp.io/smbShareAccessControl: |
      read:
        - tridentADtest
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: ontap-smb-sc
```

Exemplos e opções de configuração do ONTAP nas

Crie e utilize os drivers NAS do ONTAP com sua instalação do Trident. Esta seção fornece exemplos de configuração de backend e detalhes para mapear backends para StorageClasses. A partir da versão 25.10, o NetApp Trident oferece suporte "Sistemas de armazenamento NetApp AFX". Os sistemas de armazenamento AFX da NetApp diferem de outros sistemas baseados em ONTAP (ASA, AFF e FAS) na implementação de sua camada de storage.

OBSERVAÇÃO	Apenas o `ontap-nas` O driver (com protocolo NFS) é compatível com sistemas NetApp AFX; o protocolo SMB não é compatível.

Opções de configuração de back-end

Na configuração do backend do Trident, não é necessário especificar que seu sistema é um NetApp AFX sistema de storage. Ao selecionar `ontap-nas` como o `storageDriverName`, o Trident detecta automaticamente o sistema de storage AFX. Alguns parâmetros de configuração do backend não se aplicam a sistemas de storage AFX.

A partir da versão 26.06, você pode habilitar o posicionamento balanceado do ONTAP para que o ONTAP selecione o agregado para novos volumes. O posicionamento balanceado é necessário para sistemas de storage AFX no ONTAP 9.19.1 ou posterior. Consulte "Utilize o posicionamento balanceado com os backends ONTAP" para requisitos e exemplos.

A tabela a seguir exibe as opções de configuração do backend:

Parâmetro	Descrição	Padrão
`version`		Sempre 1

Parâmetro	Descrição	Padrão
`storageDriverName`	Nome do driver de armazenamento. Não pode ser modificado após a criação. Para atualizar este parâmetro, você deve criar um novo backend. **OBSERVAÇÃO** Somente para sistemas NetApp AFX `ontap-nas` é suportado.	`ontap-nas`, `ontap-nas-economy`, ou `ontap-nas-flexgroup`
`backendName`	Nome personalizado ou back-end de storage	Nome do driver e dataLIF
`managementLIF`	Endereço IP de um cluster ou LIF de gerenciamento de SVM Um nome de domínio totalmente qualificado (FQDN) pode ser especificado. Pode ser definido para usar endereços IPv6 se o Trident tiver sido instalado usando o sinalizador IPv6. Os endereços IPv6 devem ser definidos entre colchetes, como `[28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]` . Para o switchover MetroCluster otimizado, consulte o Exemplo de MetroCluster.	"10,0.0,1", "[2001:1234:abcd::fefe]"
`dataLIF`	Endereço IP do protocolo LIF. A NetApp recomenda `dataLIF` especificar . Se não for fornecido, o Trident buscará os dados LIFs do SVM. Você pode especificar um nome de domínio totalmente qualificado (FQDN) a ser usado para as operações de montagem NFS, permitindo que você crie um DNS round-robin para balanceamento de carga entre vários dataLIFs. Pode ser alterado após a definição inicial. Consulte a . Pode ser definido para usar endereços IPv6 se o Trident tiver sido instalado usando o sinalizador IPv6. Os endereços IPv6 devem ser definidos entre colchetes, como `[28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]` . **Omita para MetroCluster.** Consulte Exemplo de MetroCluster.	Endereço especificado ou derivado do SVM, se não for especificado (não recomendado)
`svm`	Máquina virtual de armazenamento para usar **omit for MetroCluster.** Consulte Exemplo de MetroCluster.	Derivado se uma SVM `managementLIF` for especificada
`autoExportPolicy`	Ativar a criação e atualização automática da política de exportação [Boolean]. Usando as `autoExportPolicy` opções e `autoExportCIDRs`, o Trident pode gerenciar políticas de exportação automaticamente.	falso
`autoExportCIDRs`	Lista de CIDR para filtrar IPs de nós do Kubernetes quando `autoExportPolicy` está ativado. Usando as `autoExportPolicy` opções e `autoExportCIDRs`, o Trident pode gerenciar políticas de exportação automaticamente.	["0,0.0,0/0", "::/0"]»

Parâmetro	Descrição	Padrão
`labels`	Conjunto de rótulos arbitrários formatados em JSON para aplicar em volumes	""
`clientCertificate`	Valor codificado em base64 do certificado do cliente. Usado para autenticação baseada em certificado	""
`clientPrivateKey`	Valor codificado em base64 da chave privada do cliente. Usado para autenticação baseada em certificado	""
`trustedCACertificate`	Valor codificado em base64 do certificado CA confiável. Opcional. Usado para autenticação baseada em certificado	""
`username`	Nome de usuário para conectar ao cluster/SVM. Usado para autenticação baseada em credenciais. Para autenticação do Active Directory, consulte "Autenticar o Trident em um SVM de backend usando credenciais do Active Directory".	
`password`	Senha para conectar ao cluster/SVM. Usado para autenticação baseada em credenciais. Para autenticação do Active Directory, consulte "Autenticar o Trident em um SVM de backend usando credenciais do Active Directory".	
`storagePrefix`	Prefixo usado ao provisionar novos volumes na SVM. Não pode ser atualizado depois de configurá-lo **OBSERVAÇÃO** Ao usar o ontap-nas-economy e um storagePrefix com 24 ou mais caracteres, as qtrees não têm o storage prefix incorporado, embora ele esteja presente no nome do volume.	"Trident"

Parâmetro	Descrição	Padrão
`aggregate`	Agregado para provisionamento (opcional; se definido, deve ser atribuído à SVM). Para o `ontap-nas-flexgroup` driver, esta opção é ignorada. Se não for atribuído, qualquer um dos agregados disponíveis pode ser usado para provisionar um FlexGroup volume. **OBSERVAÇÃO**: Quando o agregado é atualizado no SVM, ele é atualizado automaticamente no Trident por meio de polling no SVM, sem a necessidade de reiniciar o Trident Controller. Quando você configurou um agregado específico no Trident para provisionar volumes, se o agregado for renomeado ou movido para fora do SVM, o backend é movido para o estado de falha no Trident durante o polling do agregado no SVM. Você deve alterar o agregado para um que esteja presente no SVM ou removê-lo completamente para que o backend volte a ficar online. **Não especifique para sistemas de armazenamento AFX.** **Não especifique quando `useBalancedPlacement` estiver definido como `true`.**	""
`useBalancedPlacement`	Delegar a seleção de aggregate para novos volumes ao ONTAP [Booleano]. Requer ONTAP 9.19.1 ou posterior e `useREST` definido como `true`. Não pode ser combinado com `aggregate`. Consulte "Utilize o posicionamento balanceado com os backends ONTAP" para obter requisitos e exemplos. **Se especificado, defina sempre como `true` para sistemas de storage AFX no ONTAP 9.19.1 ou posterior.**	`false`; `true` para sistemas de armazenamento AFX no ONTAP 9.19.1 ou posterior

Parâmetro	Descrição	Padrão
`limitAggregateUsage`	O provisionamento falhará se a utilização for superior a esta percentagem. **Não se aplica ao Amazon FSx para ONTAP**. **Não especifique para sistemas de armazenamento AFX.**	"" (não aplicado por padrão)
FlexgroupAggregateList	Lista de agregados para provisionamento (opcional; se definida, deve ser atribuída ao SVM). Todos os agregados atribuídos ao SVM são usados para provisionar um volume FlexGroup. Suportado para o driver de armazenamento **ONTAP-nas-FlexGroup**. **OBSERVAÇÃO**: Quando a lista de agregados é atualizada no SVM, a lista é atualizada automaticamente no Trident por meio de consultas ao SVM, sem a necessidade de reiniciar o Trident Controller. Quando você configurou uma lista de agregados específica no Trident para provisionar volumes, se a lista de agregados for renomeada ou movida para fora do SVM, o backend será movido para o estado de falha no Trident durante a consulta à lista de agregados do SVM. Você deve alterar a lista de agregados para uma que esteja presente no SVM ou removê-la completamente para que o backend volte a ficar online.	""
`limitVolumeSize`	O provisionamento falhará se o tamanho do volume solicitado for superior a este valor.	"" (não aplicado por padrão)
`debugTraceFlags`	Debug flags para usar ao solucionar problemas. Por exemplo, não use `debugTraceFlags` a menos que você esteja solucionando problemas e exija um despejo de log detalhado.	nulo
`nasType`	Configure a criação de volumes NFS ou SMB. As opções são `nfs` , `smb` ou nulo. Definir como nulo utiliza, por padrão, volumes NFS. **Se especificado, sempre defina como `nfs` para sistemas de armazenamento AFX**.	`nfs`

Parâmetro	Descrição	Padrão
`nfsMountOptions`	Lista separada por vírgulas de opções de montagem NFS. As opções de montagem para volumes persistentes do Kubernetes normalmente são especificadas nas classes de armazenamento, mas se nenhuma opção de montagem for especificada em uma classe de armazenamento, o Trident usará as opções de montagem especificadas no arquivo de configuração do backend de storage. Se nenhuma opção de montagem for especificada na classe de armazenamento ou no arquivo de configuração, o Trident não definirá nenhuma opção de montagem em um volume persistente associado.	""
`qtreesPerFlexvol`	Qtrees máximos por FlexVol, têm de estar no intervalo [50, 300]	"200"
`smbShare`	Você pode especificar uma das seguintes opções: O nome de um compartilhamento SMB criado usando o Console de Gerenciamento da Microsoft ou a CLI do ONTAP; um nome para permitir que o Trident crie o compartilhamento SMB; ou você pode deixar o parâmetro em branco para impedir o acesso comum ao compartilhamento a volumes. Esse parâmetro é opcional para o ONTAP no local. Esse parâmetro é necessário para backends do Amazon FSX for ONTAP e não pode ficar em branco.	`smb-share`
`useREST`	Parâmetro booleano para usar APIs REST do ONTAP `.useREST`Quando definido para `true` O Trident usa APIs REST do ONTAP para se comunicar com o backend; quando configurado para `false` O Trident utiliza chamadas ONTAPI (ZAPI) para se comunicar com o backend. Este recurso requer o ONTAP 9.11.1 e posterior. Além disso, a função de login do ONTAP utilizada deve ter acesso ao `ontapi` aplicativo. Isso é satisfeito pelo predefinido `vsadmin` e `cluster-admin` papéis. A partir da versão Trident 24.06 e do ONTAP 9.15.1 ou posterior, `useREST` está definido para `true` por padrão; alterar `useREST` para `false` para usar chamadas ONTAPI (ZAPI). **Se especificado, sempre defina como `true` para sistemas de armazenamento AFX**.	`true` Para ONTAP 9.15,1 ou posterior, caso contrário `false`.
`limitVolumePoolSize`	Tamanho máximo de FlexVol requestable ao usar Qtrees no back-end ONTAP-nas-Economy.	"" (não aplicado por padrão)
`denyNewVolumePools`	Restringe `ontap-nas-economy` backends de criar novos volumes do FlexVol para conter suas Qtrees. Somente Flexvols pré-existentes são usados para provisionar novos PVS.	

Parâmetro	Descrição	Padrão
adAdminUser	Usuário ou grupo de usuários administrador do Active Directory com acesso total aos compartilhamentos SMB. Use este parâmetro para conceder direitos de administrador ao compartilhamento SMB com controle total.	

Opções de configuração de back-end para volumes de provisionamento

Você pode controlar o provisionamento padrão usando essas opções na `defaults` seção da configuração. Para obter um exemplo, consulte os exemplos de configuração abaixo.

Parâmetro	Descrição	Padrão
spaceAllocation	Alocação de espaço para Qtrees	"verdadeiro"
spaceReserve	Modo de reserva de espaço; "nenhum" (fino) ou "volume" (grosso)	"nenhum"
snapshotPolicy	Política de instantâneos a utilizar	"nenhum"
qosPolicy	Grupo de políticas de QoS a atribuir aos volumes criados. Escolha uma das qosPolicy ou adaptiveQosPolicy por pool de armazenamento/backend	""
adaptiveQosPolicy	Grupo de políticas de QoS adaptável a atribuir para volumes criados. Escolha uma das qosPolicy ou adaptiveQosPolicy por pool de armazenamento/backend. Não suportado pela ONTAP-nas-Economy.	""
snapshotReserve	Porcentagem de volume reservado para snapshots	"0" se `snapshotPolicy` for "nenhum", caso contrário ""
splitOnClone	Divida um clone de seu pai na criação	"falso"
encryption	Habilite a NetApp Volume Encryption (NVE) no novo volume; o padrão é `false`. A NVE deve estar licenciada e habilitada no cluster para usar esta opção. Se a NAE estiver habilitada no backend, qualquer volume provisionado no Trident terá a NAE habilitada. Para mais informações, consulte: "Como o Trident funciona com NVE e NAE".	"falso"
tieringPolicy	Política de disposição em camadas para usar "nenhuma"	
unixPermissions	Modo para novos volumes	"777" para volumes NFS; vazio (não aplicável) para volumes SMB
snapshotDir	Controla o acesso ao `.snapshot` diretório	`true`, `false` (Definido explicitamente).
exportPolicy	Política de exportação a utilizar	"predefinição"

Parâmetro	Descrição	Padrão
`securityStyle`	Estilo de segurança para novos volumes. Estilos de segurança e `unix` suporte de NFS `mixed`. Suporta SMB `mixed` e `ntfs` estilos de segurança.	O padrão NFS é `unix`. O padrão SMB é `ntfs`.
`nameTemplate`	Modelo para criar nomes de volume personalizados.	""

OBSERVAÇÃO	O uso de grupos de política de QoS com Trident requer o ONTAP 9.8 ou posterior. Você deve usar um grupo de políticas de QoS não compartilhado e garantir que o grupo de políticas seja aplicado individualmente a cada componente. Um grupo de políticas de QoS compartilhado impõe o limite máximo da taxa de transferência total de todos os workloads.

Exemplos de provisionamento de volume

Aqui está um exemplo com padrões definidos:

```
---
version: 1
storageDriverName: ontap-nas
backendName: customBackendName
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
labels:
  k8scluster: dev1
  backend: dev1-nasbackend
svm: trident_svm
username: cluster-admin
password: <password>
limitAggregateUsage: 80%
limitVolumeSize: 50Gi
nfsMountOptions: nfsvers=4
debugTraceFlags:
  api: false
  method: true
defaults:
  spaceReserve: volume
  qosPolicy: premium
  exportPolicy: myk8scluster
  snapshotPolicy: default
  snapshotReserve: "10"
```

Para `ontap-nas` e `ontap-nas-flexgroups` O Trident agora usa um novo cálculo para garantir que o FlexVol seja dimensionado corretamente com a porcentagem snapshotReserve e o PVC. Quando o usuário solicita um PVC, o Trident cria o FlexVol original com mais espaço usando o novo cálculo. Esse cálculo garante que o usuário receba o espaço gravável que solicitou no PVC, e não menos espaço do que o solicitado. Antes da versão 21.07, quando o usuário solicita um PVC (por exemplo, 5 GiB), com o

snapshotReserve em 50%, ele obtém apenas 2,5 GiB de espaço gravável. Isso ocorre porque o que o usuário solicitou foi o volume completo e `snapshotReserve` é uma porcentagem disso. Com o Trident 21.07, o que o usuário solicita é o espaço gravável e o Trident define o `snapshotReserve` número como porcentagem do volume total. Isso não se aplica a `ontap-nas-economy` . Veja o exemplo a seguir para ver como isso funciona:

O cálculo é o seguinte:

```
Total volume size = <PVC requested size> / (1 - (<snapshotReserve
percentage> / 100))
```

Para snapshotReserve = 50% e solicitação de PVC = 5 GiB, o tamanho total do volume é 5/.5 = 10 GiB e o tamanho disponível é 5 GiB, que é o que o usuário solicitou na solicitação de PVC . `volume show` o comando deve mostrar resultados semelhantes a este exemplo:



Os backends existentes de instalações anteriores provisionam volumes conforme explicado acima ao atualizar o Trident. Para volumes que você criou antes da atualização, você deve redimensionar esses volumes para que a alteração seja observada. Por exemplo, um PVC de 2 GiB com `snapshotReserve=50` anteriormente resultava em um volume que fornece 1 GiB de espaço gravável. Redimensionar o volume para 3 GiB, por exemplo, fornece ao aplicativo 3 GiB de espaço gravável em um volume de 6 GiB.

Exemplos mínimos de configuração

Os exemplos a seguir mostram configurações básicas que deixam a maioria dos parâmetros padrão. Esta é a maneira mais fácil de definir um backend.

OBSERVAÇÃO	Se você estiver usando o Amazon FSX no NetApp ONTAP com Trident, a recomendação é especificar nomes DNS para LIFs em vez de endereços IP.

▼ Exemplo de economia nas do ONTAP

```
---
version: 1
storageDriverName: ontap-nas-economy
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
svm: svm_nfs
username: vsadmin
password: password
```

▼ Exemplo de ONTAP nas FlexGroup

```
---
version: 1
storageDriverName: ontap-nas-flexgroup
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
svm: svm_nfs
username: vsadmin
password: password
```

▼ **Exemplo de MetroCluster**

Você pode configurar o back-end para evitar ter que atualizar manualmente a definição do back-end após o switchover e o switchback durante "Replicação e recuperação da SVM"o .

Para comutação e switchback contínuos, especifique o SVM usando `managementLIF` e omite os `dataLIF` parâmetros e. `svm` Por exemplo:

```
---
version: 1
storageDriverName: ontap-nas
managementLIF: 192.168.1.66
username: vsadmin
password: password
```

▼ **Exemplo de volumes SMB**

```
---
version: 1
backendName: ExampleBackend
storageDriverName: ontap-nas
managementLIF: 10.0.0.1
nasType: smb
securityStyle: ntfs
unixPermissions: ""
dataLIF: 10.0.0.2
svm: svm_nfs
username: vsadmin
password: password
```

▼ **Exemplo de autenticação baseada em certificado**

Este é um exemplo de configuração de back-end mínimo. `clientCertificate`, `clientPrivateKey` E `trustedCACertificate` (opcional, se estiver usando CA confiável) são preenchidos `backend.json` e recebem os valores codificados em base64 do certificado do cliente, da chave privada e do certificado de CA confiável, respetivamente.

```
---
version: 1
backendName: DefaultNASBackend
storageDriverName: ontap-nas
managementLIF: 10.0.0.1
dataLIF: 10.0.0.15
svm: nfs_svm
clientCertificate: ZXR0ZXJwYXB...ICMgJ3BhcGVyc2
clientPrivateKey: vciwKIyAgZG...0cnksIGRlc2NyaX
trustedCACertificate: zcyBbaG...b3Igb3duIGNsYXNz
storagePrefix: myPrefix_
```

▼ **Exemplo de política de exportação automática**

Este exemplo mostra como você pode instruir o Trident a usar políticas de exportação dinâmicas para criar e gerenciar a política de exportação automaticamente. Isso funciona da mesma forma para os `ontap-nas-economy` drivers e `ontap-nas-flexgroup`.

```
---
version: 1
storageDriverName: ontap-nas
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
svm: svm_nfs
labels:
  k8scluster: test-cluster-east-1a
  backend: test1-nasbackend
autoExportPolicy: true
autoExportCIDRs:
- 10.0.0.0/24
username: admin
password: password
nfsMountOptions: nfsvers=4
```

▼ **Exemplo de colocação equilibrada**

Este exemplo habilita o posicionamento balanceado do ONTAP, de forma que o ONTAP selecione o agregado para novos volumes. O posicionamento balanceado requer o ONTAP 9.19.1 ou posterior. Consulte "Utilize o posicionamento balanceado com os backends ONTAP" para obter detalhes.

```
---
version: 1
storageDriverName: ontap-nas
backendName: nas-balanced-placement
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
```

```
svm: svm_nfs
username: vsadmin
password: password
useBalancedPlacement: true
```

▼ **Exemplo de endereços IPv6**

Este exemplo mostra `managementLIF` usando um endereço IPv6.

```
---
version: 1
storageDriverName: ontap-nas
backendName: nas_ipv6_backend
managementLIF: "[5c5d:5edf:8f:7657:bef8:109b:1b41:d491]"
labels:
  k8scluster: test-cluster-east-1a
  backend: test1-ontap-ipv6
svm: nas_ipv6_svm
username: vsadmin
password: password
```

▼ **Exemplo do Amazon FSX para ONTAP usando volumes SMB**

O `smbShare` parâmetro é necessário para o FSX for ONTAP usando volumes SMB.

```
---
version: 1
backendName: SMBBackend
storageDriverName: ontap-nas
managementLIF: example.mgmt.fqdn.aws.com
nasType: smb
dataLIF: 10.0.0.15
svm: nfs_svm
smbShare: smb-share
clientCertificate: ZXR0ZXJwYXB...ICMgJ3BhcGVyc2
clientPrivateKey: vciwKIyAgZG...0cnksIGRlc2NyaX
trustedCACertificate: zcyBbaG...b3Igb3duIGNsYXNz
storagePrefix: myPrefix_
```

▼ **Exemplo de configuração de backend com nameTemplate**

```
---
version: 1
storageDriverName: ontap-nas
backendName: ontap-nas-backend
managementLIF: <ip address>
```

```
svm: svm0
username: <admin>
password: <password>
defaults:
  nameTemplate:
"{{.volume.Name}}_{{.labels.cluster}}_{{.volume.Namespace}}_{{.vo\
    lume.RequestName}}"
labels:
  cluster: ClusterA
  PVC: "{{.volume.Namespace}}_{{.volume.RequestName}}"
```

Exemplos de backends com pools virtuais

Nos arquivos de definição de back-end de exemplo mostrados abaixo, padrões específicos são definidos para todos os pools de armazenamento, como `spaceReserve` em nenhum, `spaceAllocation` em falso e `encryption` em falso. Os pools virtuais são definidos na seção armazenamento.

O Trident define rótulos de provisionamento no campo "Comentários". Os comentários são definidos no FlexVol for `ontap-nas` ou no FlexGroup `ontap-nas-flexgroup` for . O Trident copia todas as etiquetas presentes em um pool virtual para o volume de storage no provisionamento. Por conveniência, os administradores de storage podem definir rótulos por pool virtual e volumes de grupo por rótulo.

Nesses exemplos, alguns dos pools de armazenamento definem seus próprios `spaceReserve` `spaceAllocation` valores , e `encryption` , e alguns pools substituem os valores padrão.

▼ **Exemplo de ONTAP nas**

```
---
version: 1
storageDriverName: ontap-nas
managementLIF: 10.0.0.1
svm: svm_nfs
username: admin
password: <password>
nfsMountOptions: nfsvers=4
defaults:
  spaceReserve: none
  encryption: "false"
  qosPolicy: standard
labels:
  store: nas_store
  k8scluster: prod-cluster-1
region: us_east_1
storage:
  - labels:
      app: msoffice
      cost: "100"
    zone: us_east_1a
```

```
    defaults:
      spaceReserve: volume
      encryption: "true"
      unixPermissions: "0755"
      adaptiveQosPolicy: adaptive-premium
  - labels:
      app: slack
      cost: "75"
    zone: us_east_1b
    defaults:
      spaceReserve: none
      encryption: "true"
      unixPermissions: "0755"
  - labels:
      department: legal
      creditpoints: "5000"
    zone: us_east_1b
    defaults:
      spaceReserve: none
      encryption: "true"
      unixPermissions: "0755"
  - labels:
      app: wordpress
      cost: "50"
    zone: us_east_1c
    defaults:
      spaceReserve: none
      encryption: "true"
      unixPermissions: "0775"
  - labels:
      app: mysqldb
      cost: "25"
    zone: us_east_1d
    defaults:
      spaceReserve: volume
      encryption: "false"
      unixPermissions: "0775"
```

▼ **Exemplo de ONTAP nas FlexGroup**

```
---
version: 1
storageDriverName: ontap-nas-flexgroup
managementLIF: 10.0.0.1
svm: svm_nfs
username: vsadmin
```

```
password: <password>
defaults:
  spaceReserve: none
  encryption: "false"
labels:
  store: flexgroup_store
  k8scluster: prod-cluster-1
region: us_east_1
storage:
  - labels:
      protection: gold
      creditpoints: "50000"
    zone: us_east_1a
    defaults:
      spaceReserve: volume
      encryption: "true"
      unixPermissions: "0755"
  - labels:
      protection: gold
      creditpoints: "30000"
    zone: us_east_1b
    defaults:
      spaceReserve: none
      encryption: "true"
      unixPermissions: "0755"
  - labels:
      protection: silver
      creditpoints: "20000"
    zone: us_east_1c
    defaults:
      spaceReserve: none
      encryption: "true"
      unixPermissions: "0775"
  - labels:
      protection: bronze
      creditpoints: "10000"
    zone: us_east_1d
    defaults:
      spaceReserve: volume
      encryption: "false"
      unixPermissions: "0775"
```

▼ **Exemplo de economia nas do ONTAP**

```
---
version: 1
```

```
storageDriverName: ontap-nas-economy
managementLIF: 10.0.0.1
svm: svm_nfs
username: vsadmin
password: <password>
defaults:
  spaceReserve: none
  encryption: "false"
labels:
  store: nas_economy_store
region: us_east_1
storage:
  - labels:
      department: finance
      creditpoints: "6000"
    zone: us_east_1a
    defaults:
      spaceReserve: volume
      encryption: "true"
      unixPermissions: "0755"
  - labels:
      protection: bronze
      creditpoints: "5000"
    zone: us_east_1b
    defaults:
      spaceReserve: none
      encryption: "true"
      unixPermissions: "0755"
  - labels:
      department: engineering
      creditpoints: "3000"
    zone: us_east_1c
    defaults:
      spaceReserve: none
      encryption: "true"
      unixPermissions: "0775"
  - labels:
      department: humanresource
      creditpoints: "2000"
    zone: us_east_1d
    defaults:
      spaceReserve: volume
      encryption: "false"
      unixPermissions: "0775"
```

Mapeie os backends para StorageClasses

As seguintes definições de StorageClass referem-se a Exemplos de backends com pools virtuais. Usando o campo `parameters.selector`, cada StorageClass especifica quais pools virtuais podem ser usados para hospedar um volume. O volume possui os aspectos definidos no pool virtual escolhido.

- O `protection-gold` StorageClass corresponde ao primeiro e ao segundo pool virtual no `ontap-nas-flexgroup` backend. Esses são os únicos pools que oferecem proteção de nível ouro.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-gold
provisioner: csi.trident.netapp.io
parameters:
  selector: "protection=gold"
  fsType: "ext4"
```

- O `protection-not-gold` StorageClass corresponde ao terceiro e quarto pool virtual no `ontap-nas-flexgroup` backend. Estes são os únicos pools que oferecem um nível de proteção diferente do gold.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-not-gold
provisioner: csi.trident.netapp.io
parameters:
  selector: "protection!=gold"
  fsType: "ext4"
```

- O `app-mysqldb` StorageClass corresponde ao quarto pool virtual no `ontap-nas` backend. Este é o único pool que oferece configuração de pool de storage para aplicativo do tipo mysqldb.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: app-mysqldb
provisioner: csi.trident.netapp.io
parameters:
  selector: "app=mysqldb"
  fsType: "ext4"
```

- O `protection-silver-creditpoints-20k` StorageClass corresponde ao terceiro pool virtual no `ontap-nas-flexgroup` backend. Este é o único pool que oferece proteção de nível prata e 20000 pontos de crédito.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-silver-creditpoints-20k
provisioner: csi.trident.netapp.io
parameters:
  selector: "protection=silver; creditpoints=20000"
  fsType: "ext4"
```

- O `creditpoints-5k` StorageClass corresponde ao terceiro pool virtual no `ontap-nas` backend e ao segundo pool virtual no `ontap-nas-economy` backend. Essas são as únicas ofertas de pool com 5000 creditpoints.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: creditpoints-5k
provisioner: csi.trident.netapp.io
parameters:
  selector: "creditpoints=5000"
  fsType: "ext4"
```

O Trident decide qual pool virtual será selecionado e garante que o requisito de storage seja atendido.

Atualização `dataLIF` após a configuração inicial

Você pode alterar o dataLIF após a configuração inicial executando o seguinte comando para fornecer o novo arquivo JSON de back-end com dataLIF atualizado.

```
tridentctl update backend <backend-name> -f <path-to-backend-json-file-
with-updated-dataLIF>
```

OBSERVAÇÃO	Se os PVCs estiverem anexados a um ou vários pods, você deverá reduzir todos os pods correspondentes e restaurá-los para que o novo dataLIF entre em vigor.

Exemplos de SMB seguros

Configuração de backend com driver ontap-nas

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-nas
  namespace: trident
```

```
spec:
  version: 1
  storageDriverName: ontap-nas
  managementLIF: 10.0.0.1
  svm: svm2
  nasType: smb
  defaults:
    adAdminUser: tridentADtest
  credentials:
    name: backend-tbc-ontap-invest-secret
```

Configuração de backend com driver ontap-nas-economy

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-nas
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-nas-economy
  managementLIF: 10.0.0.1
  svm: svm2
  nasType: smb
  defaults:
    adAdminUser: tridentADtest
  credentials:
    name: backend-tbc-ontap-invest-secret
```

Configuração de backend com pool de armazenamento

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-nas
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-nas
  managementLIF: 10.0.0.1
  svm: svm0
  useREST: false
  storage:
  - labels:
```

```
      app: msoffice
    defaults:
      adAdminUser: tridentADuser
  nasType: smb
  credentials:
    name: backend-tbc-ontap-invest-secret
```

Exemplo de classe de armazenamento com driver ontap-nas

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-smb-sc
  annotations:
    trident.netapp.io/smbShareAdUserPermission: change
    trident.netapp.io/smbShareAdUser: tridentADtest
parameters:
  backendType: ontap-nas
  csi.storage.k8s.io/node-stage-secret-name: smbcreds
  csi.storage.k8s.io/node-stage-secret-namespace: trident
  trident.netapp.io/nasType: smb
provisioner: csi.trident.netapp.io
reclaimPolicy: Delete
volumeBindingMode: Immediate
```

OBSERVAÇÃO	Certifique-se de adicionar `annotations` para habilitar o SMB seguro. O SMB seguro não funciona sem as anotações, independentemente das configurações definidas no Backend ou no PVC.

Exemplo de classe de armazenamento com driver ontap-nas-economy

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-smb-sc
  annotations:
    trident.netapp.io/smbShareAdUserPermission: change
    trident.netapp.io/smbShareAdUser: tridentADuser3
parameters:
  backendType: ontap-nas-economy
  csi.storage.k8s.io/node-stage-secret-name: smbcreds
  csi.storage.k8s.io/node-stage-secret-namespace: trident
  trident.netapp.io/nasType: smb
provisioner: csi.trident.netapp.io
```

```
reclaimPolicy: Delete
volumeBindingMode: Immediate
```

Exemplo de PVC com um único usuário do AD

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-pvc4
  namespace: trident
  annotations:
    trident.netapp.io/smbShareAccessControl: |
      change:
        - tridentADtest
      read:
        - tridentADuser
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: ontap-smb-sc
```

Exemplo de PVC com vários usuários do AD

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-test-pvc
  annotations:
    trident.netapp.io/smbShareAccessControl: |
      full_control:
        - tridentTestuser
        - tridentuser
        - tridentTestuser1
        - tridentuser1
      change:
        - tridentADuser
        - tridentADuser1
        - tridentADuser4
        - tridentTestuser2
      read:
        - tridentTestuser2
```

```
        - tridentTestuser3
        - tridentADuser2
        - tridentADuser3
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
```

Utilize o posicionamento balanceado com os backends ONTAP

Você pode habilitar o posicionamento balanceado do ONTAP nos backends do ONTAP. Com o posicionamento balanceado, o ONTAP seleciona o agregado ideal para cada novo volume, então você não precisa gerenciar a atribuição de agregados no Trident.

Como funciona o posicionamento equilibrado

Por padrão, o Trident descobre os agregados atribuídos a uma SVM, trata cada agregado como um pool de storage e seleciona um agregado ao provisionar um volume.

Ao ativar o posicionamento equilibrado:

- O Trident reporta toda a SVM como um único pool de storage, em vez de reportar agregados individuais.
- O ONTAP seleciona o agregado para cada novo volume e equilibra os volumes entre os agregados disponíveis.
- O Trident cria volumes usando a ONTAP `/application/containers` API REST.

O posicionamento balanceado está disponível para sistemas ONTAP unificados (AFF, FAS e ASA) e NetApp AFX storage systems.

Requisitos

- O backend deve usar o ONTAP 9.19.1 ou posterior.
- O Trident deve se comunicar com o ONTAP usando API REST. O posicionamento balanceado não está disponível com ZAPI.
- A configuração do backend não deve especificar agregados. Você não pode combinar posicionamento balanceado com seleção explícita de agregados.

Valores padrão

O valor padrão de `useBalancedPlacement` depende do sistema de storage:

Sistema de storage	Padrão	Notas
O ONTAP (AFF, FAS, ASA)	`false`	O Trident seleciona agregados, a menos que você habilite o posicionamento balanceado.

Sistema de storage	Padrão	Notas
NetApp AFX (ONTAP 9.19.1 ou posterior)	`true`	É necessário um posicionamento equilibrado. Definir `useBalancedPlacement` para `false` causa falha na validação do backend.
NetApp AFX (anterior ao ONTAP 9.19.1)	`false`	A colocação balanceada não está disponível.

Opção de configuração

Parâmetro	Descrição	Padrão
`useBalancedPlacement`	Delegar a seleção de agregados para novos volumes ao ONTAP. Defina esta opção na raiz da configuração do backend ou em pools virtuais individuais.	`false`; `true` para AFX no ONTAP 9.19.1 ou posterior

Alocação equilibrada com pools virtuais

O sistema de alocação equilibrada funciona com ou sem pools virtuais:

- Sem pools virtuais, o Trident reporta o SVM como um único pool de storage e cria todos os novos volumes usando posicionamento balanceado.
- Com pools virtuais, o Trident reporta os pools virtuais e associa os volumes a eles normalmente. Se o pool virtual correspondente tiver o posicionamento balanceado ativado, o ONTAP seleciona o agregado para o volume.

Defina `useBalancedPlacement` na raiz da configuração do backend para aplicá-la a todos os pools virtuais ou defina em pools virtuais individuais para habilitar seletivamente.

Exemplos

O exemplo a seguir permite o posicionamento balanceado em um backend ONTAP unificado:

```
---
version: 1
storageDriverName: ontap-nas
backendName: nas-balanced-placement
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
svm: trident_svm
username: cluster-admin
password: <password>
useBalancedPlacement: true
```

O exemplo a seguir permite o posicionamento equilibrado usando `TridentBackendConfig`:

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
```

```
metadata:
  name: backend-tbc-ontap-nas
spec:
  version: 1
  storageDriverName: ontap-nas
  backendName: nas-balanced-placement
  managementLIF: 10.0.0.1
  svm: trident_svm
  useBalancedPlacement: true
  credentials:
    name: backend-tbc-ontap-nas-secret
```

O exemplo a seguir permite o posicionamento balanceado no `gold` pool virtual apenas. O Trident continua selecionando agregados para volumes que correspondam ao `bronze` pool virtual:

```
---
version: 1
storageDriverName: ontap-nas
backendName: nas-virtual-pools
managementLIF: 10.0.0.1
svm: trident_svm
username: cluster-admin
password: <password>
storage:
- labels:
    performance: gold
  useBalancedPlacement: true
- labels:
    performance: bronze
```

Erros de validação no backend

A criação ou atualização do backend falha nos seguintes casos:

Configuração	Resultado
`useBalancedPlacement: true`no ONTAP anterior à versão 9.19.1	A validação falhou. Atualize para o ONTAP 9.19.1 ou posterior.
`useBalancedPlacement: true` em um backend usando ZAPI	A validação falhou. Configure o backend para usar API REST.
`useBalancedPlacement: true` combinado com agregados específicos	A validação falhou. Remova a configuração do aggregate ou desative o posicionamento balanceado.
`useBalancedPlacement: false`no AFX com o ONTAP 9.19.1 ou posterior	A validação falhou. É necessário um posicionamento balanceado para o AFX no ONTAP 9.19.1 ou posterior.

Amazon FSX para NetApp ONTAP

Use o Trident com o Amazon FSX para NetApp ONTAP

"Amazon FSX para NetApp ONTAP" é um serviço totalmente gerenciado da AWS que executa sistemas de arquivos alimentados pelo sistema operacional de storage NetApp ONTAP. Ele fornece recursos, desempenho e administração do ONTAP com a escalabilidade e simplicidade operacional da AWS. Um sistema de arquivos é o recurso principal no Amazon FSx e é análogo a um cluster ONTAP local. Cada sistema de arquivos contém uma ou mais máquinas virtuais de armazenamento (SVMs), e cada SVM contém um ou mais volumes que armazenam arquivos e diretórios. Essa integração permite que clusters Kubernetes em execução no Amazon Elastic Kubernetes Service (EKS) provisionem volumes persistentes com suporte a ONTAP para cargas de trabalho de bloco e arquivo.

Requisitos

Além "Requisitos da Trident"do , para integrar o FSX for ONTAP com o Trident, você precisa:

- Um cluster do Amazon EKS existente ou um cluster do Kubernetes autogerenciado com `kubectl` o instalado.
- Um sistema de arquivos e máquina virtual de armazenamento (SVM) do Amazon FSX for NetApp ONTAP que pode ser acessado a partir dos nós de trabalho do seu cluster.
- Nós de trabalho preparados para "NFS ou iSCSI".

OBSERVAÇÃO	Certifique-se de seguir as etapas de preparação de nós necessárias para o Amazon Linux e "Imagens de máquinas da Amazon" Ubuntu (AMIS), dependendo do seu tipo de AMI EKS.

Considerações

- Volumes SMB:
  - Os volumes SMB são suportados usando `ontap-nas` apenas o driver.
  - Os volumes SMB não são compatíveis com o complemento Trident EKS.
  - Trident suporta volumes SMB montados em pods executados apenas em nós Windows. Consulte "Prepare-se para provisionar volumes SMB" para obter detalhes.
- Antes do Trident 24,02, os volumes criados nos sistemas de arquivos do Amazon FSX que têm backups automáticos ativados, não puderam ser excluídos pelo Trident. Para evitar esse problema no Trident 24,02 ou posterior, especifique o `fsxFilesystemID`, `apiRegion` AWS , AWS `apikey` e AWS `secretKey` no arquivo de configuração de back-end do AWS FSX for ONTAP.

OBSERVAÇÃO	Se você estiver especificando uma função do IAM para o Trident, poderá omitir especificar explicitamente os `apiRegion` campos , `apiKey` e `secretKey` para o Trident. Para obter mais informações, "Opções e exemplos de configuração do FSX for ONTAP"consulte .

Uso simultâneo do driver Trident SAN/iSCSI e EBS-CSI

Se você planeja usar drivers ontap-san (por exemplo, iSCSI) com AWS (EKS, ROSA, EC2 ou qualquer outra instância), a configuração multipath necessária nos nós pode entrar em conflito com o driver CSI do Amazon Elastic Block Store (EBS). Para garantir que o multipathing funcione sem interferir nos discos EBS no mesmo nó, você precisa excluir o EBS da sua configuração de multipathing. Este exemplo mostra um `multipath.conf` arquivo que inclui as configurações necessárias do Trident , excluindo discos EBS do multipathing:

```
defaults {
    find_multipaths no
}
blacklist {
    device {
        vendor "NVME"
        product "Amazon Elastic Block Store"
    }
}
```

Autenticação

O Trident oferece dois modos de autenticação.

- Baseado em credenciais (recomendado): Armazena credenciais com segurança no AWS Secrets Manager. Você pode usar o `fsxadmin` usuário do sistema de arquivos ou o `vsadmin` usuário configurado para o SVM.

AVISO	O Trident espera ser executado como um `vsadmin` usuário SVM ou como um usuário com um nome diferente que tenha a mesma função. O Amazon FSX for NetApp ONTAP tem um `fsxadmin` usuário que é uma substituição limitada do usuário do cluster do ONTAP `admin`. Recomendamos vivamente a utilização `vsadmin` com o Trident.

- Baseado em certificado: O Trident se comunicará com o SVM em seu sistema de arquivos FSX usando um certificado instalado em seu SVM.

Para obter detalhes sobre como ativar a autenticação, consulte a autenticação do tipo de driver:

- "Autenticação nas ONTAP"
- "Autenticação SAN ONTAP"

Imagens de máquinas da Amazon testadas (AMIS)

O cluster do EKS é compatível com vários sistemas operacionais, mas a AWS otimizou determinadas AMIS (Amazon Machine Images) para contêineres e EKS. As seguintes AMIs foram testadas com o NetApp Trident 25.02.

AMI	NAS	Economia nas	ISCSI	iSCSI-economia
AL2023_x86_64_STANDARD	Sim	Sim	Sim	Sim

AL2_x86_64	Sim	Sim	Sim*	Sim*
BOTTLEROCKET_x86_64	Sim**	Sim	N/A.	N/A.
AL2023_ARM_64_STANDARD	Sim	Sim	Sim	Sim
AL2_ARM_64	Sim	Sim	Sim*	Sim*
BOTTLEROCKET_ARM_64	Sim**	Sim	N/A.	N/A.

- * Não é possível excluir o PV sem reiniciar o nó
- ** Não funciona com NFSv3 com Trident versão 25.02.

OBSERVAÇÃO	Se o seu IAM desejado não está listado aqui, isso não significa que ele não é suportado; simplesmente significa que ele não foi testado. Esta lista serve como um guia para AMIs que são conhecidas por funcionar.

Testes realizados com:

- Versão EKS: 1.32
- Método de instalação: Helm 25.06 e como um complemento AWS 25.06
- Para nas, tanto o NFSv3 quanto o NFSv4,1 foram testados.
- Para SAN, apenas o iSCSI foi testado, não o NVMe-of.

Testes realizados:

- Criar: Classe de armazenamento, pvc, pod
- Excluir: Pod, PVC (regular, qtree/lun – economia, nas com backup da AWS)

Encontre mais informações

- "Documentação do Amazon FSX para NetApp ONTAP"
- "Blog post no Amazon FSX for NetApp ONTAP"

Crie uma função do IAM e o AWS Secret

Você pode configurar pods do Kubernetes para acessar recursos da AWS autenticando como uma função do AWS IAM em vez de fornecer credenciais explícitas da AWS.

OBSERVAÇÃO	Para autenticar usando uma função do AWS IAM, você deve ter um cluster do Kubernetes implantado usando o EKS.

Crie o segredo do AWS Secrets Manager

Como o Trident estará emitindo APIs em um vserver do FSX para gerenciar o armazenamento para você, ele precisará de credenciais para fazer isso. A maneira segura de passar essas credenciais é através de um segredo do AWS Secrets Manager. Portanto, se você ainda não tiver um, precisará criar um segredo do AWS Secrets Manager que contenha as credenciais da conta vsadmin.

Este exemplo cria um segredo do AWS Secrets Manager para armazenar credenciais do Trident CSI:

```
aws secretsmanager create-secret --name trident-secret --description
"Trident CSI credentials"\
  --secret-string
"{\"username\":\"vsadmin\",\"password\":\"<svmpassword>\"}"
```

Criar política do IAM

O Trident também precisa de permissões da AWS para ser executado corretamente. Portanto, você precisa criar uma política que dê ao Trident as permissões de que ele precisa.

Os exemplos a seguir criam uma política do IAM usando a AWS CLI:

```
aws iam create-policy --policy-name AmazonFSxNCSIDriverPolicy --policy
-document file://policy.json
  --description "This policy grants access to Trident CSI to FSxN and
Secrets manager"
```

- Exemplo JSON de política*:

```
{
  "Statement": [
    {
      "Action": [
        "fsx:DescribeFileSystems",
        "fsx:DescribeVolumes",
        "fsx:CreateVolume",
        "fsx:RestoreVolumeFromSnapshot",
        "fsx:DescribeStorageVirtualMachines",
        "fsx:UntagResource",
        "fsx:UpdateVolume",
        "fsx:TagResource",
        "fsx:DeleteVolume"
      ],
      "Effect": "Allow",
      "Resource": "*"
    },
    {
      "Action": "secretsmanager:GetSecretValue",
      "Effect": "Allow",
      "Resource": "arn:aws:secretsmanager:<aws-region>:<aws-account-
id>:secret:<aws-secret-manager-name>*"
    }
  ],
```

```
  "Version": "2012-10-17"
}
```

Criar identidade de pod ou função IAM para associação de conta de serviço (IRSA)

Você pode configurar uma conta de serviço do Kubernetes para assumir uma função de Gerenciamento de Identidade e Acesso (IAM) da AWS com a função de Identidade do Pod do EKS ou a função de IAM para Associação de Conta de Serviço (IRSA). Quaisquer Pods configurados para usar a conta de serviço podem acessar qualquer serviço da AWS para o qual a função tenha permissão de acesso.

Identidade do Pod

As associações de identidade do Amazon EKS Pod fornecem a capacidade de gerenciar credenciais para seus aplicativos, de forma semelhante à maneira como os perfis de instância do Amazon EC2 fornecem credenciais para instâncias do Amazon EC2.

Instale o Pod Identity no seu cluster EKS:

Você pode criar uma identidade de Pod por meio do console da AWS ou usando o seguinte comando da AWS CLI:

```
aws eks create-addon --cluster-name <EKS_CLUSTER_NAME> --addon-name
eks-pod-identity-agent
```

Para mais informações consulte "Configurar o agente de identidade do pod do Amazon EKS" .

Crie trust-relationship.json:

Crie trust-relationship.json para permitir que o EKS Service Principal assuma essa função para a Identidade do Pod. Em seguida, crie uma função com esta política de confiança:

```
aws iam create-role \
  --role-name fsxn-csi-role --assume-role-policy-document file://trust-
relationship.json \
  --description "fsxn csi pod identity role"
```

arquivo trust-relationship.json:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "pods.eks.amazonaws.com"
      },
      "Action": [
```

```
        "sts:AssumeRole",
        "sts:TagSession"
      ]
    }
  ]
}
```

Anexe a política de função à função do IAM:

Anexe a política de função da etapa anterior à função do IAM que foi criada:

```
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:111122223333:policy/fsxn-csi-policy \
  --role-name fsxn-csi-role
```

Crie uma associação de identidade de pod:

Crie uma associação de identidade de pod entre a função do IAM e a conta de serviço do Trident (trident-controller)

```
aws eks create-pod-identity-association \
  --cluster-name <EKS_CLUSTER_NAME> \
  --role-arn arn:aws:iam::111122223333:role/fsxn-csi-role \
  --namespace trident --service-account trident-controller
```

Função do IAM para associação de conta de serviço (IRSA)

Usando a AWS CLI:

```
aws iam create-role --role-name AmazonEKS_FSxN_CSI_DriverRole \
  --assume-role-policy-document file://trust-relationship.json
```

- arquivo trust-relation.json:*

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws:iam::<account_id>:oidc-
provider/<oidc_provider>"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
```

```
      "Condition": {
        "StringEquals": {
          "<oidc_provider>:aud": "sts.amazonaws.com",
          "<oidc_provider>:sub":
"system:serviceaccount:trident:trident-controller"
        }
      }
    }
  ]
}
```

Atualize os seguintes valores no `trust-relationship.json` arquivo:

- **<account_id>** - seu ID de conta da AWS
- **<oidc_provider>** - o OIDC do seu cluster EKS. Você pode obter o oidc_provider executando:

```
aws eks describe-cluster --name my-cluster --query
"cluster.identity.oidc.issuer"\
  --output text | sed -e "s/^https:\/\///"
```

Anexar a função do IAM com a política do IAM:

Depois que a função tiver sido criada, anexe a política (que foi criada na etapa acima) à função usando este comando:

```
aws iam attach-role-policy --role-name my-role --policy-arn <IAM policy
ARN>
```

Verifique se o provedor OICD está associado:

Verifique se seu provedor de OIDC está associado ao cluster. Você pode verificá-lo usando este comando:

```
aws iam list-open-id-connect-providers | grep $oidc_id | cut -d "/" -f4
```

Se a saída estiver vazia, use o seguinte comando para associar o IAM OIDC ao cluster:

```
eksctl utils associate-iam-oidc-provider --cluster $cluster_name
--approve
```

Se você estiver usando eksctl, use o exemplo a seguir para criar uma função do IAM para uma conta de serviço no EKS:

```
eksctl create iamserviceaccount --name trident-controller --namespace
trident \
  --cluster <my-cluster> --role-name AmazonEKS_FSxN_CSI_DriverRole
--role-only \
  --attach-policy-arn <IAM-Policy ARN> --approve
```

Instale o Trident

Trident simplifica o gerenciamento de storage do Amazon FSx for NetApp ONTAP no Kubernetes para permitir que seus desenvolvedores e administradores se concentrem na implantação de aplicativos. Você pode instalar Trident usando um dos seguintes métodos:

- Leme
- Complemento EKS

Se quiser utilizar a funcionalidade de instantâneos, instale o suplemento do controlador de instantâneos CSI. "Ativar a funcionalidade de instantâneos para volumes CSI"Consulte para obter mais informações.

Instale o Trident através do leme

Identidade do Pod

1. Adicione o repositório Helm do Trident:

```
helm repo add netapp-trident https://netapp.github.io/trident-helm-
chart
```

2. Instale o Trident usando o seguinte exemplo:

```
helm install trident-operator netapp-trident/trident-operator
--version 100.2502.1 --namespace trident --create-namespace
```

Você pode usar o `helm list` comando para revisar detalhes de instalação, como nome, namespace, gráfico, status, versão do aplicativo e número de revisão.

```
helm list -n trident
```

```
NAME                NAMESPACE   REVISION    UPDATED
STATUS       CHART                          APP VERSION

trident-operator    trident     1           2024-10-14
```

```
14:31:22.463122 +0300 IDT    deployed     trident-operator-
100.2502.0    25.02.0
```

Associação de contas de serviço (IRSA)

1. Adicione o repositório Helm do Trident:

```
helm repo add netapp-trident https://netapp.github.io/trident-helm-
chart
```

2. Defina os valores para **provedor de nuvem** e **identidade da nuvem**:

```
helm install trident-operator netapp-trident/trident-operator
--version 100.2502.1  \
--set cloudProvider="AWS" \
--set cloudIdentity="'eks.amazonaws.com/role-arn:
arn:aws:iam::<accountID>:role/<AmazonEKS_FSxN_CSI_DriverRole>'" \
--namespace trident \
--create-namespace
```

Você pode usar o `helm list` comando para revisar detalhes de instalação, como nome, namespace, gráfico, status, versão do aplicativo e número de revisão.

```
helm list -n trident
```

```
NAME                NAMESPACE   REVISION    UPDATED
STATUS       CHART                          APP VERSION

trident-operator    trident     1           2024-10-14
14:31:22.463122 +0300 IDT    deployed     trident-operator-
100.2510.0    25.10.0
```

OBSERVAÇÃO

Se você planeja usar iSCSI, certifique-se de que o iSCSI esteja habilitado na sua máquina cliente. Se estiver usando o sistema operacional AL2023 Worker node, você pode automatizar a instalação do cliente iSCSI adicionando o parâmetro node prep na instalação do helm:

```
helm install trident-operator netapp-trident/trident-
operator --version 100.2502.1 --namespace trident
--create-namespace --set nodePrep={iscsi}
```

Instale o Trident através do suplemento EKS

O complemento do Trident EKS inclui os patches de segurança mais recentes, correções de bugs e é validado pela AWS para funcionar com o Amazon EKS. O complemento EKS permite que você garanta consistentemente que seus clusters do Amazon EKS estejam seguros e estáveis e reduza a quantidade de trabalho que você precisa fazer para instalar, configurar e atualizar complementos.

Pré-requisitos

Verifique se você tem o seguinte antes de configurar o complemento do Trident para o AWS EKS:

- Uma conta de cluster do Amazon EKS com assinatura complementar
- Permissões da AWS para o marketplace da AWS:
  `"aws-marketplace:ViewSubscriptions",`
  `"aws-marketplace:Subscribe",`
  `"aws-marketplace:Unsubscribe`
- Tipo de AMI: Amazon Linux 2 (AL2_x86_64) ou Amazon Linux 2 ARM(AL2_ARM_64)
- Tipo de nó: AMD ou ARM
- Um sistema de arquivos existente do Amazon FSX for NetApp ONTAP

Ative o complemento Trident para AWS

Console de gerenciamento

1. Abra o console do Amazon EKS em https://console.aws.amazon.com/eks/home#/clusters.
2. No painel de navegação esquerdo, selecione **clusters**.
3. Selecione o nome do cluster para o qual deseja configurar o complemento NetApp Trident CSI.
4. Selecione **Add-ons** e, em seguida, selecione **Get more add-ons**.
5. Siga estas etapas para selecionar o complemento:
   a. Role para baixo até a seção **Complementos do AWS Marketplace** e digite **"Trident"** na caixa de pesquisa.
   b. Marque a caixa de seleção no canto superior direito da caixa Trident by NetApp.
   c. Selecione **seguinte**.
6. Na página de configurações **Configure Selected add-ons**, faça o seguinte:

OBSERVAÇÃO	**Ignore estas etapas se estiver usando a associação de identidade do Pod.**

   a. Selecione a **versão** que você gostaria de usar.
   b. Se você estiver usando a autenticação IRSA, certifique-se de definir os valores de configuração disponíveis nas Configurações opcionais:
      - Selecione a **versão** que você gostaria de usar.
      - Siga o **Esquema de configuração do complemento** e defina o parâmetro **configurationValues** na seção **Valores de configuração** para o role-arn que você criou na etapa anterior (o valor deve estar no seguinte formato):

```
{

  "cloudIdentity": "'eks.amazonaws.com/role-arn: <role ARN>'",
  "cloudProvider": "AWS"

}
```

+
Se você selecionar Substituir para o método de resolução de conflitos, uma ou mais configurações para o suplemento existente podem ser sobrescritas com as configurações de complemento do Amazon EKS. Se você não ativar essa opção e houver um conflito com suas configurações existentes, a operação falhará. Você pode usar a mensagem de erro resultante para solucionar o conflito. Antes de selecionar essa opção, certifique-se de que o complemento do Amazon EKS não gerencie as configurações que você precisa para gerenciar automaticamente.

7. Escolha **seguinte**.
8. Na página **Revisão e adição**, escolha **criar**.

   Depois que a instalação do complemento estiver concluída, você verá o complemento instalado.

CLI DA AWS

1. Crie o `add-on.json` arquivo:

Para Identidade do Pod, use o seguinte formato:

OBSERVAÇÃO Use o

```
{
  "clusterName": "<eks-cluster>",
  "addonName": "netapp_trident-operator",
  "addonVersion": "v25.6.0-eksbuild.1",
}
```

Para autenticação IRSA, use o seguinte formato:

```
{
  "clusterName": "<eks-cluster>",
  "addonName": "netapp_trident-operator",
  "addonVersion": "v25.6.0-eksbuild.1",
  "serviceAccountRoleArn": "<role ARN>",
  "configurationValues": {
    "cloudIdentity": "'eks.amazonaws.com/role-arn: <role ARN>'",
    "cloudProvider": "AWS"
  }
}
```

OBSERVAÇÃO	Substitua `<role ARN>` pelo ARN da função criada na etapa anterior.

2. Instale o complemento Trident EKS.

```
aws eks create-addon --cli-input-json file://add-on.json
```

eksctl

O seguinte comando de exemplo instala o complemento do Trident EKS:

```
eksctl create addon --name netapp_trident-operator --cluster
<cluster_name> --force
```

Atualize o complemento Trident EKS

Console de gerenciamento

1. Abra o console do Amazon EKS https://console.aws.amazon.com/eks/home#/clusters .
2. No painel de navegação esquerdo, selecione **clusters**.
3. Selecione o nome do cluster para o qual deseja atualizar o complemento NetApp Trident CSI.
4. Selecione a guia **Complementos**.
5. Selecione **Trident by NetApp** e, em seguida, selecione **Edit**.
6. Na página **Configurar Trident by NetApp**, faça o seguinte:
   a. Selecione a **versão** que você gostaria de usar.
   b. Expanda **Configurações opcionais de configuração** e modifique conforme necessário.
   c. Selecione **Salvar alterações**.

CLI DA AWS

O exemplo a seguir atualiza o complemento EKS:

```
aws eks update-addon --cluster-name <eks_cluster_name> --addon-name
netapp_trident-operator --addon-version v25.6.0-eksbuild.1 \
  --service-account-role-arn <role-ARN> --resolve-conflict preserve \
  --configuration-values "{\"cloudIdentity\":
\"'eks.amazonaws.com/role-arn: <role ARN>'\"}"
```

eksctl

- Verifique a versão atual do seu complemento FSxN Trident CSI. Substitua `my-cluster` pelo nome do cluster.

```
eksctl get addon --name netapp_trident-operator --cluster my-cluster
```

Exemplo de saída:

```
NAME                          VERSION              STATUS    ISSUES
IAMROLE    UPDATE AVAILABLE    CONFIGURATION VALUES
netapp_trident-operator    v25.6.0-eksbuild.1    ACTIVE    0
{"cloudIdentity":"'eks.amazonaws.com/role-arn:
arn:aws:iam::139763910815:role/AmazonEKS_FSXN_CSI_DriverRole'"}
```

- Atualize o complemento para a versão retornada em ATUALIZAÇÃO DISPONÍVEL na saída da etapa anterior.

```
eksctl update addon --name netapp_trident-operator --version
v25.6.0-eksbuild.1 --cluster my-cluster --force
```

Se você remover `--force` a opção e qualquer uma das configurações de complemento do Amazon EKS entrar em conflito com as configurações existentes, a atualização do complemento do Amazon EKS falhará; você receberá uma mensagem de erro para ajudá-lo a resolver o conflito. Antes de especificar essa opção, verifique se o complemento do Amazon EKS não gerencia as configurações que você precisa gerenciar, pois essas configurações são sobrescritas com essa opção. Para obter mais informações sobre outras opções para essa configuração, "Complementos"consulte . Para obter mais informações sobre o gerenciamento de campo do Amazon EKS Kubernetes, "Gerenciamento de campo do Kubernetes"consulte .

Desinstale/remova o complemento Trident EKS

Você tem duas opções para remover um complemento do Amazon EKS:

- **Preserve o software complementar no cluster** – essa opção remove o gerenciamento do Amazon EKS de qualquer configuração. Ele também remove a capacidade do Amazon EKS de notificá-lo de atualizações e atualizar automaticamente o complemento do Amazon EKS depois de iniciar uma atualização. No entanto, ele preserva o software complementar no cluster. Essa opção torna o complemento uma instalação autogerenciada, em vez de um complemento do Amazon EKS. Com essa opção, não há tempo de inatividade para o complemento. Guarde a `--preserve` opção no comando para preservar o complemento.
- **Remover software complementar inteiramente do cluster** – a NetApp recomenda que você remova o complemento do Amazon EKS do cluster somente se não houver recursos no cluster que dependam dele. Remova `--preserve` a opção do `delete` comando para remover o complemento.

OBSERVAÇÃO	Se o complemento tiver uma conta do IAM associada a ele, a conta do IAM não será removida.

Console de gerenciamento

1. Abra o console do Amazon EKS em https://console.aws.amazon.com/eks/home#/clusters.
2. No painel de navegação esquerdo, selecione **clusters**.
3. Selecione o nome do cluster para o qual deseja remover o complemento NetApp Trident CSI.

4. Selecione a guia **Complementos** e, em seguida, selecione **Trident by NetApp**.*
5. Selecione **Remover**.
6. Na caixa de diálogo **Remover NetApp_Trident-operator confirmation**, faça o seguinte:
   a. Se você quiser que o Amazon EKS pare de gerenciar as configurações do complemento, selecione **Preserve on cluster**. Faça isso se quiser manter o software complementar no cluster para que você possa gerenciar todas as configurações do complemento por conta própria.
   b. Digite **NetApp_Trident-operator**.
   c. Selecione **Remover**.

CLI DA AWS

Substitua `my-cluster` pelo nome do cluster e execute o seguinte comando.

```
aws eks delete-addon --cluster-name my-cluster --addon-name
netapp_trident-operator --preserve
```

eksctl

O seguinte comando desinstala o complemento do Trident EKS:

```
eksctl delete addon --cluster K8s-arm --name netapp_trident-operator
```

Configure uma classe de armazenamento

O "Objeto Kubernetes StorageClass" identifica um provisionador e instrui o provisionador sobre como provisionar volumes. Esta seção mostra como configurar um objeto StorageClass do Kubernetes que especifica Trident como provisionador.

Criar um StorageClass Objeto

Ao criar um StorageClass para FSx for ONTAP, o Trident criará automaticamente a configuração de backend.

OBSERVAÇÃO	Se você deseja configurar manualmente o backend de armazenamento, consulte a Crie um Kubernetes StorageClass sem configuração automática de backend seção sobre como criar o backend Trident e a classe de armazenamento separadamente.

Especifique os parâmetros necessários de StorageClass

Os três parâmetros a seguir precisam ser definidos ao criar um StorageClass:

Parâmetro	Obrigatório	Tipo	Descrição
`fsxFilesystemID`	Sim	cadeia de carateres	FSx para NetApp ONTAP ID do sistema de arquivos
`storageDriverName`	Sim	cadeia de carateres	Driver de storage Trident (por exemplo, `ontap-nas` ou `ontap-san`)

Parâmetro	Obrigatório	Tipo	Descrição
`credentialsName`	Sim	cadeia de carateres	Nome do segredo do Kubernetes que contém as credenciais do FSx for ONTAP

Especifique parâmetros opcionais

Você pode passar parâmetros opcionais para o backend através do StorageClass. Defina todos os valores opcionais como strings na StorageClass `parameters` section. Para obter uma lista completa dos parâmetros de backend, consulte: "Configuração do backend FSx para NetApp ONTAP".

Exemplo de arquivos de configuração StorageClass.

O exemplo a seguir mostra um StorageClass que aciona a configuração automática do backend.

YAML

```yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-fsx-demo
  annotations:
    description: "Demo StorageClass for FSx for NetApp ONTAP"
provisioner: csi.trident.netapp.io
parameters:
  fsxFilesystemID: "fs-0abc123"
  storageDriverName: "ontap-nas"
  credentialsName: trident-fsx-credentials
allowVolumeExpansion: true
reclaimPolicy: Delete
volumeBindingMode: Immediate
```

JSON

```json
{
  "apiVersion": "storage.k8s.io/v1",
  "kind": "StorageClass",
  "metadata": {
    "name": "ontap-fsx-demo",
    "annotations": {
      "description": "Demo StorageClass for FSx for NetApp ONTAP"
    }
  },
  "provisioner": "csi.trident.netapp.io",
  "parameters": {
    "fsxFilesystemID": "fs-0abc123",
    "storageDriverName": "ontap-nas",
```

```
    "credentialsName": "trident-fsx-credentials"
  },
  "allowVolumeExpansion": true,
  "reclaimPolicy": "Delete",
  "volumeBindingMode": "Immediate"
}
```

Crie o StorageClass

Após criar o arquivo de configuração, execute o seguinte comando para criar a storage class.

```
kubectl create -f storage-class-ontapnas.yaml
```

Agora você deve ver uma classe de armazenamento **Basic-csi** no Kubernetes e no Trident, e o Trident deve ter descoberto os pools no back-end.

```
kubectl get sc basic-csi
```

```
NAME         PROVISIONER             AGE
basic-csi    csi.trident.netapp.io   15h
```

Após aplicar o StorageClass, Trident cria o backend automaticamente. Você pode então criar PersistentVolumeClaims que referenciam esse StorageClass.

Verifique o status da configuração do backend

Trident registra o resultado da criação do backend em anotações de StorageClass.

Anotação	Descrição
`trident.netapp.io/configuratorStatus`	Resultado da configuração (`Success` ou `Failure`)
`trident.netapp.io/configuratorMessage`	Mensagem detalhada de status ou erro
`trident.netapp.io/configuratorName`	Nome do recurso interno do configurador
`trident.netapp.io/managed`	Indica que o StorageClass é gerenciado pela Trident
`trident.netapp.io/additionalStoragePools`	Pools de storage criados para este backend

Para verificar o status, execute:

```
kubectl get storageclass ontap-fsx-demo -o yaml
```

Confirme que `trident.netapp.io/configuratorStatus` está definido como `Success`. Se o valor for

`Failure`, revise `trident.netapp.io/configuratorMessage` para o erro.

Adicionar sistemas de arquivos FSxN adicionais

Se você precisar de capacidade de storage adicional enquanto continua usando o mesmo StorageClass, adicione IDs de sistema de arquivos FSxN adicionais.

Edite o StorageClass e adicione a seguinte anotação:

```
metadata:
  annotations:
    trident.netapp.io/additionalFsxNFileSystemID: '["fs-
xxxxxxxxxxxxxxxxxx"]'
```

Após aplicar a alteração, Trident atualiza a configuração do backend e atualiza as anotações de StorageClass.

Considerações operacionais e limitações

- Excluir um StorageClass que possui a configuração automática de backend geralmente exclui o Trident backend associado. Isso pode interromper a conectividade de storage e interromper cargas de trabalho em execução. Valide o impacto antes de excluir um StorageClass gerenciado.
- A configuração automática de backend é suportada apenas para AWS FSx para NetApp ONTAP.

Crie um Kubernetes StorageClass sem configuração automática de backend

Se você deseja criar o backend do Trident e o StorageClass separadamente, siga estas etapas.

Entenda como funciona a configuração automática do backend

Trident deriva a configuração do backend da definição de StorageClass. Quando você aplica o StorageClass, Trident valida os parâmetros necessários, cria o backend e anota o StorageClass com o status.

Trident cria o VolumeSnapshotClass apenas uma vez. Trident reutiliza o mesmo VolumeSnapshotClass para StorageClasses subsequentes.

Crie o backend do Trident

Para criar um backend Trident, você precisa criar um arquivo de configuração em formato JSON ou YAML. O arquivo deve especificar o tipo de storage desejado (NAS ou SAN), o sistema de arquivos, a SVM de onde obtê-lo e como autenticar com ela. O exemplo a seguir mostra como definir um storage baseado em NAS e usar um segredo da AWS para armazenar as credenciais da SVM que você deseja usar:

YAML

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-nas
  namespace: trident
```

```
spec:
  version: 1
  storageDriverName: ontap-nas
  backendName: tbc-ontap-nas
  svm: svm-name
  aws:
    fsxFilesystemID: fs-xxxxxxxxxx
  credentials:
    name: "arn:aws:secretsmanager:us-west-2:xxxxxxxx:secret:secret-
name"
    type: awsarn
```

JSON

```
{
  "apiVersion": "trident.netapp.io/v1",
  "kind": "TridentBackendConfig",
  "metadata": {
    "name": "backend-tbc-ontap-nas"
    "namespace": "trident"
  },
  "spec": {
    "version": 1,
    "storageDriverName": "ontap-nas",
    "backendName": "tbc-ontap-nas",
    "svm": "svm-name",
    "aws": {
      "fsxFilesystemID": "fs-xxxxxxxxxx"
    },
    "managementLIF": null,
    "credentials": {
      "name": "arn:aws:secretsmanager:us-west-2:xxxxxxxx:secret:secret-
name",
      "type": "awsarn"
    }
  }
}
```

Detalhes do driver FSX for ONTAP

Você pode integrar o Trident com o Amazon FSX for NetApp ONTAP usando os seguintes drivers:

Nome do driver	Descrição
`ontap-san`	Cada PV provisionado é um LUN dentro de seu próprio volume Amazon FSx for NetApp ONTAP. Recomendado para armazenamento em bloco.
`ontap-nas`	Cada PV provisionado é um volume completo do Amazon FSx for NetApp ONTAP. Recomendado para NFS e SMB.
`ontap-san-economy`	Cada PV provisionado é um LUN com um número configurável de LUNs por Amazon FSx for NetApp ONTAP volume.
`ontap-nas-economy`	Cada PV provisionado é uma qtree, com um número configurável de qtrees por volume do Amazon FSx for NetApp ONTAP.
`ontap-nas-flexgroup`	Cada PV provisionado é um volume completo do Amazon FSx for NetApp ONTAP FlexGroup.

Para obter informações sobre o condutor, "Controladores NAS"consulte e "Controladores SAN".

Crie o backend

Após criar o arquivo de configuração, execute os seguintes comandos para criar e validar a Trident Backend Configuration (TBC):

- Crie a configuração de back-end do Trident (TBC) a partir do arquivo yaml e execute o seguinte comando:

```
kubectl create -f backendconfig.yaml -n trident
```

```
tridentbackendconfig.trident.netapp.io/backend-tbc-ontap-nas created
```

- Validar a configuração de back-end do Trident (TBC) foi criada com sucesso:

```
Kubectl get tbc -n trident
```

```
NAME                          BACKEND NAME          BACKEND UUID
PHASE   STATUS

backend-tbc-ontap-nas         tbc-ontap-nas         933e0071-66ce-4324-
b9ff-f96d916ac5e9   Bound   Success
```

Para obter mais informações sobre outras opções de configuração, consulte a Configuração avançada do backend e exemplos seção abaixo.

Configurar uma Storage Class sem configuração automática de backend

A seguir, apresentamos exemplos de configurações de Storage Class para uso com Trident e FSx for ONTAP.

Classe de armazenamento para NFS

Você pode usar este exemplo para configurar StorageClass para volumes usando NFS (consulte a seção de Atributos do Trident abaixo para obter a lista completa de atributos):

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-gold
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-nas"
  provisioningType: "thin"
  snapshots: "true"
```

Classe de armazenamento para iSCSI

Use este exemplo para configurar StorageClass para volumes usando iSCSI:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-gold
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
  provisioningType: "thin"
  snapshots: "true"
```

Classe de armazenamento usando NFSv3 e AWS Bottlerocket

Para provisionar volumes NFSv3 no AWS Bottlerocket, adicione o necessário `mountOptions` à classe de armazenamento:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-gold
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-nas"
  media: "ssd"
  provisioningType: "thin"
  snapshots: "true"
mountOptions:
```

```
  - nfsvers=3
  - nolock
```

Atributos do Trident StorageClass

Esses parâmetros determinam quais pools de storage gerenciado pelo Trident devem ser utilizados para provisionar volumes de um determinado tipo.

Atributo	Tipo	Valores	Oferta	Pedido	Suportado por
1	cadeia de carateres	hdd, híbrido, ssd	Pool contém Mídia desse tipo; híbrido significa ambos	Tipo de material especificado	ONTAP-nas, ONTAP-nas-economy, ONTAP-nas-FlexGroup, ONTAP-san, SolidFire-san
ProvisioningType	cadeia de carateres	fino, grosso	O pool é compatível com esse método de provisionamento	Método de provisionamento especificado	thick: all ONTAP; thin: all ONTAP & SolidFire-san
BackendType	cadeia de carateres	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, solidfire-san, azure-netapp-files, ontap-san-economy	Pool pertence a este tipo de backend	Back-end especificado	Todos os drivers
instantâneos	bool	verdadeiro, falso	O pool é compatível com volumes com snapshots	Volume com instantâneos ativados	ontap-nas, ontap-san, solidfire-san
clones	bool	verdadeiro, falso	O pool é compatível com volumes de clonagem	Volume com clones ativados	ontap-nas, ontap-san, solidfire-san
criptografia	bool	verdadeiro, falso	O pool é compatível com volumes criptografados	Volume com encriptação ativada	ONTAP-nas, ONTAP-nas-economy, ONTAP-nas-flexgroups, ONTAP-san
IOPS	int	número inteiro positivo	O pool é capaz de garantir IOPS nessa faixa	Volume garantido estas operações de entrada/saída por segundo	SolidFire-san

[1]: Não suportado pelo ONTAP Select ou FSx for ONTAP systems

"Objetos Kubernetes e Trident"Consulte para obter detalhes sobre como as classes de armazenamento interagem com os `PersistentVolumeClaim` parâmetros e para controlar como o Trident provisiona volumes.

Crie a classe de armazenamento

Depois de configurar o StorageClass, você pode criá-lo no Kubernetes.

Passos

1. Esse é um objeto do Kubernetes, então use `kubectl` para criá-lo no Kubernetes.

```
kubectl create -f storage-class-ontapnas.yaml
```

2. Agora você deve ver uma classe de armazenamento **Basic-csi** no Kubernetes e no Trident, e o Trident deve ter descoberto os pools no back-end.

```
kubectl get sc basic-csi
```

```
NAME         PROVISIONER             AGE
basic-csi    csi.trident.netapp.io   15h
```

Provisionar volumes SMB

Você pode provisionar volumes SMB usando o `ontap-nas` driver. No entanto, para isso, você deve concluir estas etapas: "Prepare-se para provisionar volumes SMB".

Configuração avançada do backend e exemplos

Consulte a tabela a seguir para obter as opções de configuração de back-end:

Parâmetro	Descrição	Exemplo
`version`		Sempre 1
`storageDriverName`	Nome do controlador de armazenamento	`ontap-nas` `ontap-nas-economy`, , `ontap-nas-flexgroup` `ontap-san` , , , `ontap-san-economy`
`backendName`	Nome personalizado ou back-end de storage	Nome do driver e dataLIF

Parâmetro	Descrição	Exemplo
`managementLIF`	Endereço IP de um cluster ou LIF de gerenciamento de SVM Um nome de domínio totalmente qualificado (FQDN) pode ser especificado. Pode ser definido para usar endereços IPv6 se o Trident tiver sido instalado usando o sinalizador IPv6. Os endereços IPv6 devem ser definidos entre colchetes, como [28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]. Se você fornecer o `fsxFilesystemID` sob o `aws` campo, não precisará fornecer o `managementLIF` porque o Trident recupera as informações do SVM `managementLIF` da AWS. Portanto, você deve fornecer credenciais para um usuário sob o SVM (por exemplo: Vsadmin) e o usuário deve ter a `vsadmin` função.	"10,0.0,1", "[2001:1234:abcd::fefe]"
`dataLIF`	Endereço IP do protocolo LIF. **ONTAP NAS drivers**: NetApp recomenda especificar dataLIF. Caso não seja fornecido, Trident busca os dataLIFs no SVM. Você pode especificar um nome de domínio totalmente qualificado (FQDN) para ser usado nas operações de montagem NFS, permitindo criar um DNS round-robin para balancear a carga entre vários dataLIFs. Pode ser alterado após a configuração inicial. **ONTAP SAN drivers**: Não especifique para iSCSI. Trident usa ONTAP Selective LUN Map para descobrir os LIFs iSCSI necessários para estabelecer uma sessão multipath. Um aviso é gerado se dataLIF for definido explicitamente. Pode ser configurado para usar endereços IPv6 se Trident foi instalado usando o parâmetro IPv6. Os endereços IPv6 devem ser definidos entre colchetes, como [28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555].	

Parâmetro	Descrição	Exemplo
`autoExportPolicy`	Ativar a criação e atualização automática da política de exportação [Boolean]. Usando as `autoExportPolicy` opções e `autoExportCIDRs`, o Trident pode gerenciar políticas de exportação automaticamente.	`false`
`autoExportCIDRs`	Lista de CIDR para filtrar IPs de nós do Kubernetes quando `autoExportPolicy` está ativado. Usando as `autoExportPolicy` opções e `autoExportCIDRs`, o Trident pode gerenciar políticas de exportação automaticamente.	"["0,0.0,0/0", ":::/0"]"
`labels`	Conjunto de rótulos arbitrários formatados em JSON para aplicar em volumes	""
`clientCertificate`	Valor codificado em base64 do certificado do cliente. Usado para autenticação baseada em certificado	""
`clientPrivateKey`	Valor codificado em base64 da chave privada do cliente. Usado para autenticação baseada em certificado	""
`trustedCACertificate`	Valor codificado em base64 do certificado CA confiável. Opcional. Usado para autenticação baseada em certificado.	""
`username`	Nome de usuário para se conetar ao cluster ou SVM. Usado para autenticação baseada em credenciais. Por exemplo, vsadmin.	
`password`	Senha para se conectar ao cluster ou SVM. Usado para autenticação baseada em credenciais.	
`svm`	Máquina virtual de armazenamento para usar	Derivado se um SVM managementLIF for especificado.
`storagePrefix`	Prefixo usado ao provisionar novos volumes na SVM. Não pode ser modificado após a criação. Para atualizar esse parâmetro, você precisará criar um novo backend.	`trident`

Parâmetro	Descrição	Exemplo
`limitAggregateUsage`	**Não especifique para o Amazon FSX for NetApp ONTAP.** O fornecido `fsxadmin` e `vsadmin` não contém as permissões necessárias para recuperar o uso agregado e limitá-lo usando o Trident.	Não utilizar.
`limitVolumeSize`	Falha no provisionamento se o tamanho do volume solicitado estiver acima desse valor. Também restringe o tamanho máximo dos volumes que gerencia para qtrees e LUNs, e a `qtreesPerFlexvol` opção permite personalizar o número máximo de qtrees por FlexVol volume	"" (não aplicado por padrão)
`lunsPerFlexvol`	O máximo de LUNs por FlexVol volume tem de estar no intervalo [50, 200]. Apenas SAN.	"100"
`debugTraceFlags`	Debug flags para usar ao solucionar problemas. Por exemplo, não use `debugTraceFlags` a menos que você esteja solucionando problemas e exija um despejo de log detalhado.	nulo
`nfsMountOptions`	Lista separada por vírgulas de opções de montagem NFS. As opções de montagem para volumes persistentes do Kubernetes normalmente são especificadas em classes de armazenamento, mas se nenhuma opção de montagem for especificada em uma classe de armazenamento, o Trident voltará a usar as opções de montagem especificadas no arquivo de configuração do back-end de armazenamento. Se nenhuma opção de montagem for especificada na classe de armazenamento ou no arquivo de configuração, o Trident não definirá nenhuma opção de montagem em um volume persistente associado.	""

Parâmetro	Descrição	Exemplo
`nasType`	Configurar a criação de volumes NFS ou SMB. As opções são `nfs`, `smb`, ou null. **Deve definir como `smb` para volumes SMB.** A configuração como null padrão para volumes NFS.	`nfs`
`qtreesPerFlexvol`	Qtrees máximos por FlexVol volume, têm de estar no intervalo [50, 300]	`"200"`
`smbShare`	Você pode especificar uma das seguintes opções: O nome de um compartilhamento SMB criado usando o Console de Gerenciamento da Microsoft ou a CLI do ONTAP ou um nome para permitir que o Trident crie o compartilhamento SMB. Esse parâmetro é necessário para backends do Amazon FSX for ONTAP.	`smb-share`
`useREST`	Parâmetro booleano para usar APIs REST do ONTAP. Quando definido como `true`, o Trident usará APIs REST do ONTAP para se comunicar com o back-end. Esse recurso requer o ONTAP 9.11,1 e posterior. Além disso, a função de login do ONTAP usada deve ter acesso ao `ontap` aplicativo. Isso é satisfeito com as funções e `cluster-admin` predefinidas `vsadmin`.	`false`
`aws`	Você pode especificar o seguinte no arquivo de configuração do AWS FSX for ONTAP: - `fsxFilesystemID`: Especifique o ID do sistema de arquivos AWS FSX. `apiRegion`- : Nome da região da API AWS. `apikey`- : Chave da API da AWS. `secretKey`- : Chave secreta da AWS.	`""` `""` `""`

Parâmetro	Descrição	Exemplo
`credentials`	Especifique as credenciais do FSX SVM para armazenar no AWS Secrets Manager. `name`- : Nome do recurso Amazon (ARN) do segredo, que contém as credenciais do SVM. `type`- : Defina para `awsarn`. "Crie um segredo do AWS Secrets Manager"Consulte para obter mais informações.	

Opções de configuração de back-end para volumes de provisionamento

Você pode controlar o provisionamento padrão usando essas opções na `defaults` seção da configuração. Para obter um exemplo, consulte os exemplos de configuração abaixo.

Parâmetro	Descrição	Padrão
`spaceAllocation`	Alocação de espaço para LUNs	`true`
`spaceReserve`	Modo de reserva de espaço; "nenhum" (fino) ou "volume" (grosso)	`none`
`snapshotPolicy`	Política de instantâneos a utilizar	`none`
`qosPolicy`	Grupo de políticas de QoS a atribuir aos volumes criados. Escolha uma das qosPolicy ou adaptiveQosPolicy por pool de armazenamento ou backend. O uso de grupos de política de QoS com Trident requer o ONTAP 9.8 ou posterior. Você deve usar um grupo de políticas de QoS não compartilhado e garantir que o grupo de políticas seja aplicado individualmente a cada componente. Um grupo de políticas de QoS compartilhado impõe o limite máximo da taxa de transferência total de todos os workloads.	""
`adaptiveQosPolicy`	Grupo de políticas de QoS adaptável a atribuir para volumes criados. Escolha uma das qosPolicy ou adaptiveQosPolicy por pool de armazenamento ou backend. Não suportado pela ONTAP-nas-Economy.	""
`snapshotReserve`	Percentagem de volume reservado para instantâneos "0"	Se `snapshotPolicy` for `none`, `else` ""

Parâmetro	Descrição	Padrão
`splitOnClone`	Divida um clone de seu pai na criação	`false`
`encryption`	Ative a criptografia de volume do NetApp (NVE) no novo volume; o padrão é `false`. O NVE deve ser licenciado e habilitado no cluster para usar essa opção. Se NAE estiver ativado no back-end, qualquer volume provisionado no Trident será NAE habilitado. Para obter mais informações, consulte: "Como o Trident funciona com NVE e NAE".	`false`
`luksEncryption`	Ative a criptografia LUKS. Consulte "Usar a configuração de chave unificada do Linux (LUKS)". Somente SAN.	""
`tieringPolicy`	Política de disposição em camadas para usar `none`	
`unixPermissions`	Modo para novos volumes. **Deixe vazio para volumes SMB.**	""
`securityStyle`	Estilo de segurança para novos volumes. Estilos de segurança e `unix` suporte de NFS `mixed`. Suporta SMB `mixed` e `ntfs` estilos de segurança.	O padrão NFS é `unix`. O padrão SMB é `ntfs`.

Configurar um PVC

Esta seção inclui instruções sobre como criar um PersistentVolumeClaim (PVC) que usa o StorageClass do Kubernetes configurado para solicitar um PV. Se a solicitação for bem-sucedida, você poderá montar o PV em um pod.

Crie o PVC

Um "*PersistentVolumeClaim*" (PVC) é uma solicitação de acesso ao PersistentVolume no cluster. O PVC pode ser configurado para solicitar armazenamento de um determinado tamanho ou modo de acesso. Usando o StorageClass associado, o administrador do cluster pode controlar mais do que apenas o tamanho e o modo de acesso do PersistentVolume—como desempenho ou nível de serviço.

Após criar o backend Trident e o StorageClass, você pode criar um PVC. Depois que o PVC for criado, você pode montar o volume em um pod.

Manifestos de amostra

Os exemplos a seguir mostram opções básicas de configuração de PVC.

PVC com acesso RWX

Este exemplo mostra um PVC básico com acesso RWX associado a um StorageClass `basic-csi` chamado .

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc-storage
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: ontap-gold
```

Exemplo de PVC usando iSCSI

Este exemplo mostra um PVC básico para iSCSI com acesso RWO que está associado a uma StorageClass denominada `protection-gold` .

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc-san
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: protection-gold
```

Criar PVC

Passos

1. Crie o PVC.

   ```
   kubectl create -f pvc.yaml
   ```

2. Verifique o estado do PVC.

   ```
   kubectl get pvc
   ```

```
NAME        STATUS VOLUME     CAPACITY ACCESS MODES STORAGECLASS AGE
```

```
pvc-storage Bound  pv-name 2Gi      RWO                          5m
```

"Objetos Kubernetes e Trident"Consulte para obter detalhes sobre como as classes de armazenamento interagem com os `PersistentVolumeClaim` parâmetros e para controlar como o Trident provisiona volumes.

Implantar um aplicativo

Quando a classe de armazenamento e PVC são criados, você pode montar o PV em um pod. Esta seção lista o comando de exemplo e a configuração para anexar o PV a um pod.

Implante um aplicativo de exemplo

Passos

1. Monte o volume num pod.

```
kubectl create -f pv-pod.yaml
```

Estes exemplos mostram configurações básicas para anexar o PVC a um pod: **Configuração básica**:

```
kind: Pod
apiVersion: v1
metadata:
  name: pv-pod
spec:
  volumes:
    - name: pv-storage
      persistentVolumeClaim:
       claimName: basic
  containers:
    - name: pv-container
      image: nginx
      ports:
        - containerPort: 80
          name: "http-server"
      volumeMounts:
        - mountPath: "/my/mount/path"
          name: pv-storage
```

OBSERVAÇÃO	Pode monitorizar o progresso utilizando `kubectl get pod --watch`o .

2. Verifique se o volume está montado no `/my/mount/path`.

```
kubectl exec -it pv-pod -- df -h /my/mount/path
```

```
Filesystem                                                      Size
Used  Avail Use%  Mounted on
192.168.188.78:/trident_pvc_ae45ed05_3ace_4e7c_9080_d2a83ae03d06  1.1G
320K  1.0G   1%   /my/mount/path
```

Agora você pode excluir o Pod. O aplicativo Pod não existirá mais, mas o volume permanecerá.

```
kubectl delete pod pv-pod
```

Configure o complemento do Trident EKS em um cluster EKS

O NetApp Trident simplifica o gerenciamento de armazenamento do Amazon FSX for NetApp ONTAP no Kubernetes para permitir que seus desenvolvedores e administradores se concentrem na implantação de aplicativos. O complemento do NetApp Trident EKS inclui os patches de segurança mais recentes, correções de bugs e é validado pela AWS para funcionar com o Amazon EKS. O complemento EKS permite que você garanta consistentemente que seus clusters do Amazon EKS estejam seguros e estáveis e reduza a quantidade de trabalho que você precisa fazer para instalar, configurar e atualizar complementos.

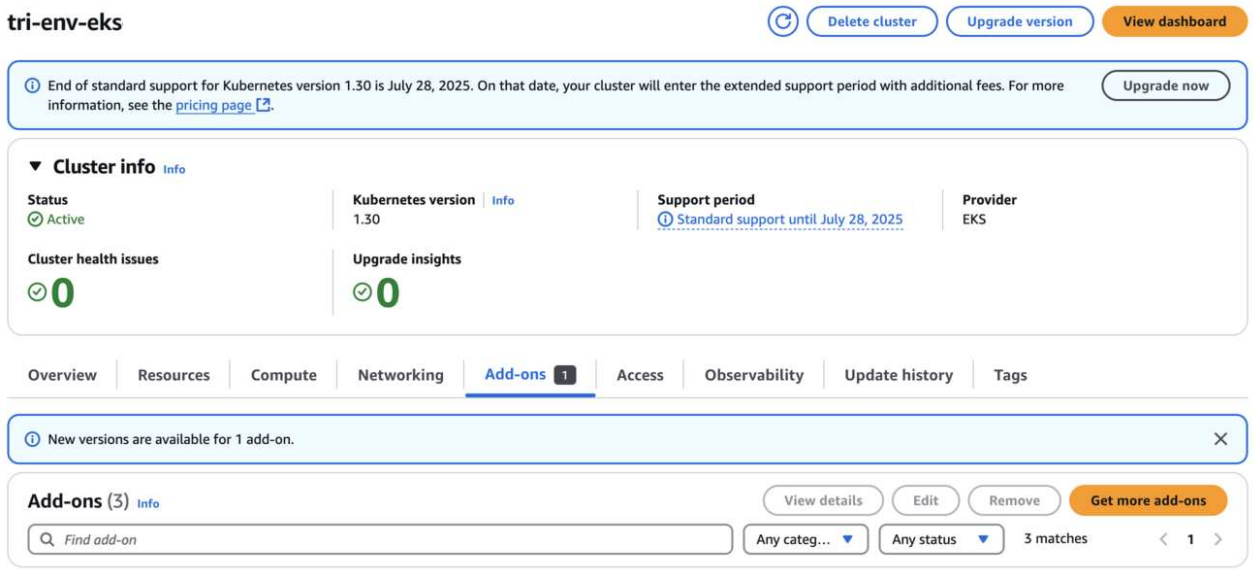
Pré-requisitos

Verifique se você tem o seguinte antes de configurar o complemento do Trident para o AWS EKS:

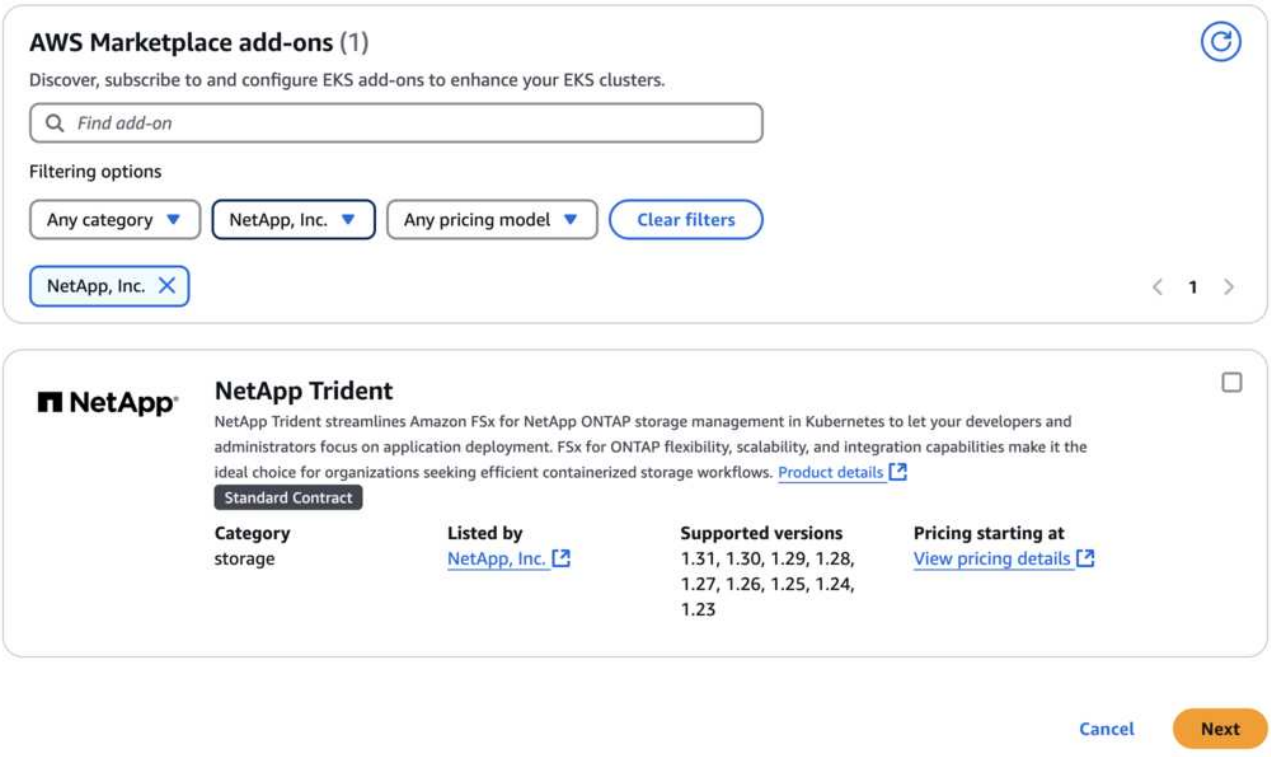
- Uma conta de cluster do Amazon EKS com permissões para trabalhar com complementos. "Complementos do Amazon EKS"Consulte a .
- Permissões da AWS para o marketplace da AWS:
  `"aws-marketplace:ViewSubscriptions",`
  `"aws-marketplace:Subscribe",`
  `"aws-marketplace:Unsubscribe`
- Tipo de AMI: Amazon Linux 2 (AL2_x86_64) ou Amazon Linux 2 ARM(AL2_ARM_64)
- Tipo de nó: AMD ou ARM
- Um sistema de arquivos existente do Amazon FSX for NetApp ONTAP

Passos

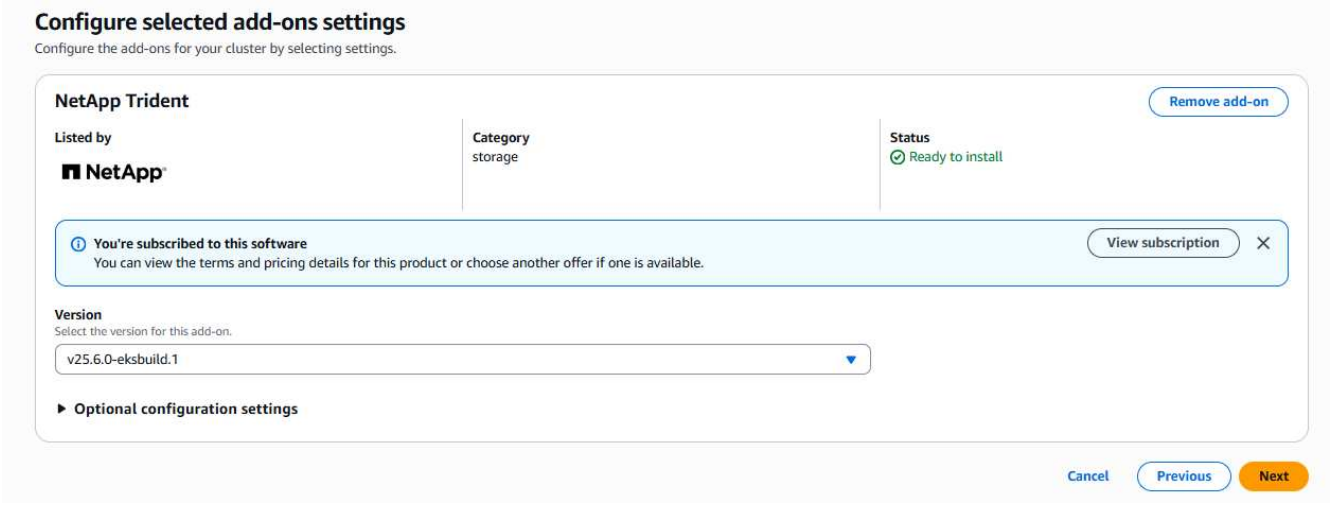
1. Certifique-se de criar a função do IAM e o segredo da AWS para permitir que os pods do EKS acessem recursos da AWS. Para obter instruções, "Crie uma função do IAM e o AWS Secret"consulte .
2. No cluster do EKS Kubernetes, navegue até a guia **Complementos**.



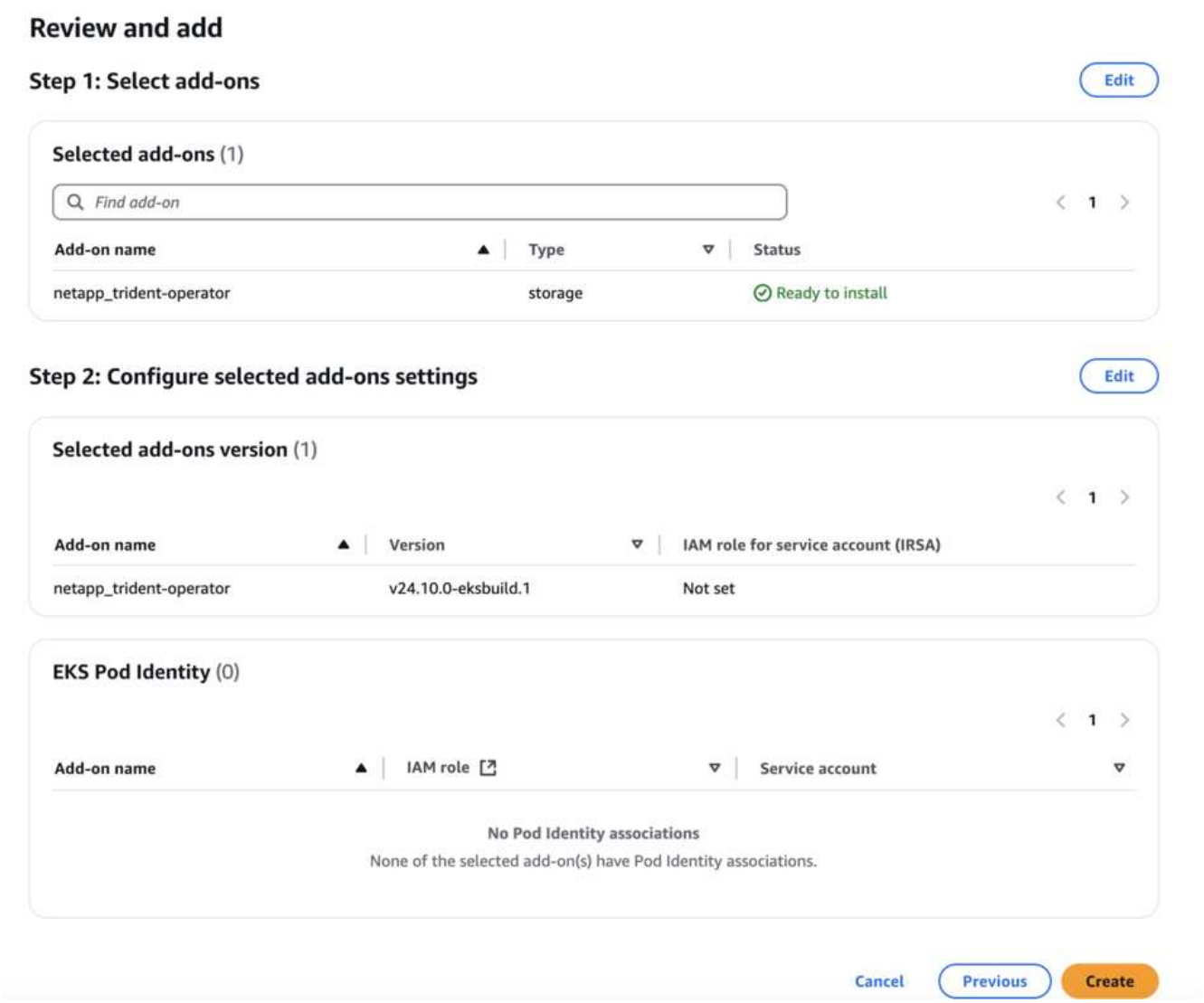
3. Vá para **Complementos do AWS Marketplace** e escolha a categoria *storage*.



4. Localize **NetApp Trident** e marque a caixa de seleção do complemento Trident e clique em **Avançar**.
5. Escolha a versão desejada do complemento.

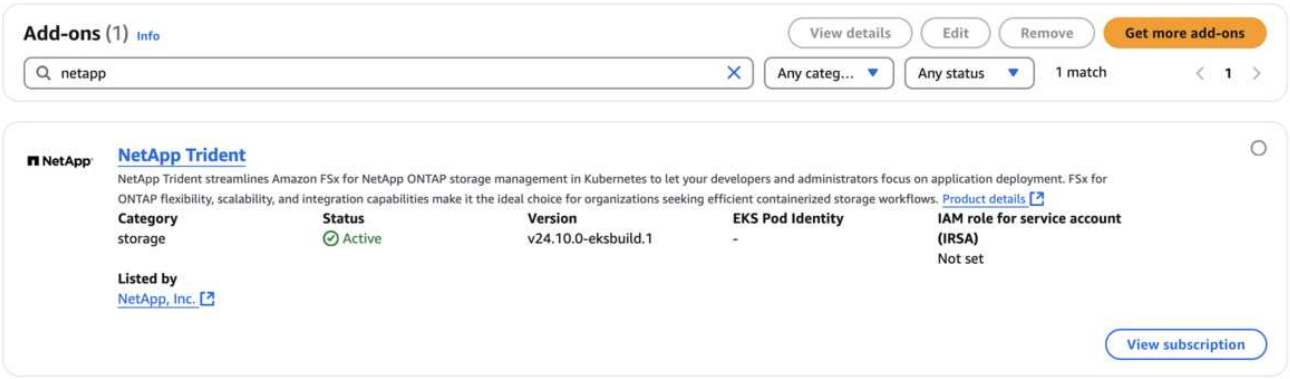


6. Configure as configurações necessárias do complemento.



7. Se você estiver usando IRSA (funções do IAM para conta de serviço), consulte as etapas de configuração adicionais"aqui" .
8. Selecione **criar**.

9. Verifique se o status do complemento é *ative*.



10. Execute o seguinte comando para verificar se o Trident está instalado corretamente no cluster:

```
kubectl get pods -n trident
```

11. Continue a configuração e configure o backend de storage. Para obter informações, consulte "Configure o back-end de armazenamento".

Instale/desinstale o complemento Trident EKS usando a CLI

Instale o complemento NetApp Trident EKS usando CLI:

O comando de exemplo a seguir instala o complemento Trident EKS: `eksctl create addon --cluster clusterName --name netapp_trident-operator --version v25.6.0-eksbuild.1` (com uma versão dedicada)

O seguinte comando de exemplo instala o Trident EKS add-on versão 25.6.1: `eksctl create addon --cluster clusterName --name netapp_trident-operator --version v25.6.1-eksbuild.1` (com uma versão dedicada)

O seguinte comando de exemplo instala o Trident EKS add-on versão 25.6.2: `eksctl create addon --cluster clusterName --name netapp_trident-operator --version v25.6.2-eksbuild.1` (com uma versão dedicada)

Desinstale o complemento NetApp Trident EKS usando a CLI:

O seguinte comando desinstala o complemento do Trident EKS:

```
eksctl delete addon --cluster K8s-arm --name netapp_trident-operator
```

Crie backends com kubectl

Um back-end define a relação entre o Trident e um sistema de storage. Ele informa à Trident como se comunicar com esse sistema de storage e como o Trident deve provisionar volumes a partir dele. Após a instalação do Trident, a próxima etapa é criar um backend. A `TridentBackendConfig` Definição de recursos personalizada (CRD) permite criar e gerenciar backends Trident diretamente por meio da interface do

Kubernetes. Para fazer isso, use `kubectl` ou a ferramenta CLI equivalente para sua distribuição do Kubernetes.

`TridentBackendConfig`

`TridentBackendConfig`(`tbc tbconfig, , tbackendconfig`) É um CRD com namespaces e frontend que permite gerenciar backends Trident usando `kubectl`o . Agora, os administradores de storage e Kubernetes podem criar e gerenciar back-ends diretamente pela CLI do Kubernetes sem exigir um utilitário de linha de comando dedicado (`tridentctl`).

Após a criação de `TridentBackendConfig` um objeto, acontece o seguinte:

- Um back-end é criado automaticamente pelo Trident com base na configuração que você fornece. Isto é representado internamente como um `TridentBackend` (`tbe`, `tridentbackend`) CR.
- O `TridentBackendConfig` é exclusivamente vinculado a um `TridentBackend` que foi criado por Trident.

Cada `TridentBackendConfig` um mantém um mapeamento um-para-um com um `TridentBackend`. o primeiro é a interface fornecida ao usuário para projetar e configurar backends; o último é como o Trident representa o objeto backend real.

AVISO	`TridentBackend` Os CRS são criados automaticamente pelo Trident. Você **não deve** modificá-los. Se você quiser fazer atualizações para backends, faça isso modificando o `TridentBackendConfig` objeto.

Veja o exemplo a seguir para o formato do `TridentBackendConfig` CR:

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-san
spec:
  version: 1
  backendName: ontap-san-backend
  storageDriverName: ontap-san
  managementLIF: 10.0.0.1
  dataLIF: 10.0.0.2
  svm: trident_svm
  credentials:
    name: backend-tbc-ontap-san-secret
```

Você também pode dar uma olhada nos exemplos "instalador do Trident" no diretório para configurações de exemplo para a plataforma/serviço de armazenamento desejado.

O `spec` utiliza parâmetros de configuração específicos do back-end. Neste exemplo, o backend usa o `ontap-san` driver de armazenamento e usa os parâmetros de configuração que são tabulados aqui. Para obter a lista de opções de configuração para o driver de armazenamento desejado, consulte o "informações de configuração de back-end para seu driver de armazenamento".

A `spec` seção também inclui `credentials` campos e `deletionPolicy`, que são recentemente introduzidos no `TridentBackendConfig` CR:

- `credentials`: Este parâmetro é um campo obrigatório e contém as credenciais usadas para autenticar com o sistema/serviço de armazenamento. Isso é definido como um segredo do Kubernetes criado pelo usuário. As credenciais não podem ser passadas em texto simples e resultarão em um erro.
- `deletionPolicy`: Este campo define o que deve acontecer quando o `TridentBackendConfig` é excluído. Pode tomar um dos dois valores possíveis:
  - `delete`: Isso resulta na exclusão do `TridentBackendConfig` CR e do back-end associado. Este é o valor padrão.
  - `retain`: Quando um `TridentBackendConfig` CR é excluído, a definição de back-end ainda estará presente e poderá ser gerenciada com `tridentctl`o . Definir a política de exclusão para `retain` permitir que os usuários façam o downgrade para uma versão anterior (anterior a 21,04) e mantenham os backends criados. O valor para este campo pode ser atualizado após a criação de um `TridentBackendConfig`.

OBSERVAÇÃO	O nome de um back-end é definido usando `spec.backendName`. Se não for especificado, o nome do backend é definido como o nome do `TridentBackendConfig` objeto (metadata.name). Recomenda-se definir explicitamente nomes de back-end usando `spec.backendName`o .
DICA	Backends que foram criados com `tridentctl` não têm um objeto associado `TridentBackendConfig`. Você pode optar por gerenciar esses backends `kubectl` criando um `TridentBackendConfig` CR. Deve-se ter cuidado para especificar parâmetros de configuração idênticos (como `spec.backendName` , , `spec.storagePrefix`, `spec.storageDriverName` e assim por diante). O Trident vinculará automaticamente o recém-criado `TridentBackendConfig` com o back-end pré-existente.

Visão geral dos passos

Para criar um novo back-end usando `kubectl`o , você deve fazer o seguinte:

1. Criar um "Segredo do Kubernetes". o segredo contém as credenciais que o Trident precisa para se comunicar com o cluster/serviço de storage.
2. Crie `TridentBackendConfig` um objeto. Isso contém detalhes sobre o cluster/serviço de armazenamento e faz referência ao segredo criado na etapa anterior.

Depois de criar um backend, você pode observar seu status usando `kubectl get tbc <tbc-name> -n <trident-namespace>` e coletar detalhes adicionais.

Etapa 1: Crie um segredo do Kubernetes

Crie um segredo que contenha as credenciais de acesso para o back-end. Isso é exclusivo para cada serviço/plataforma de storage. Aqui está um exemplo:

```
kubectl -n trident create -f backend-tbc-ontap-san-secret.yaml
```

```
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-ontap-san-secret
type: Opaque
stringData:
  username: cluster-admin
  password: password
```

Esta tabela resume os campos que devem ser incluídos no segredo para cada plataforma de armazenamento:

Descrição dos campos secretos da plataforma de armazenamento	Segredo	Descrição dos campos
Azure NetApp Files	ID do cliente	A ID do cliente a partir de um registo de aplicação
Elemento (NetApp HCI/SolidFire)	Endpoint	MVIP para o cluster SolidFire com credenciais de locatário
ONTAP	nome de utilizador	Nome de usuário para se conetar ao cluster/SVM. Usado para autenticação baseada em credenciais
ONTAP	palavra-passe	Senha para se conectar ao cluster/SVM. Usado para autenticação baseada em credenciais
ONTAP	ClientPrivateKey	Valor codificado em base64 da chave privada do cliente. Usado para autenticação baseada em certificado
ONTAP	ChapUsername	Nome de utilizador de entrada. Necessário se useCHAP-true. Para `ontap-san` e. `ontap-san-economy`
ONTAP	IniciadorSecreto	Segredo do iniciador CHAP. Necessário se useCHAP-true. Para `ontap-san` e. `ontap-san-economy`
ONTAP	ChapTargetUsername	Nome de utilizador alvo. Necessário se useCHAP-true. Para `ontap-san` e. `ontap-san-economy`
ONTAP	ChapTargetInitiatorSecret	Segredo do iniciador de destino CHAP. Necessário se useCHAP-true. Para `ontap-san` e. `ontap-san-economy`

O segredo criado nesta etapa será referenciado `spec.credentials` no campo do `TridentBackendConfig` objeto que é criado na próxima etapa.

Passo 2: Crie o `TridentBackendConfig` CR

Agora você está pronto para criar seu `TridentBackendConfig` CR. Neste exemplo, um back-end que usa `ontap-san` o driver é criado usando o `TridentBackendConfig` objeto mostrado abaixo:

```
kubectl -n trident create -f backend-tbc-ontap-san.yaml
```

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-san
spec:
  version: 1
  backendName: ontap-san-backend
  storageDriverName: ontap-san
  managementLIF: 10.0.0.1
  dataLIF: 10.0.0.2
  svm: trident_svm
  credentials:
    name: backend-tbc-ontap-san-secret
```

Etapa 3: Verifique o status do `TridentBackendConfig` CR

Agora que criou o `TridentBackendConfig` CR, pode verificar o estado. Veja o exemplo a seguir:

```
kubectl -n trident get tbc backend-tbc-ontap-san
NAME                     BACKEND NAME          BACKEND UUID
PHASE   STATUS
backend-tbc-ontap-san   ontap-san-backend     8d24fce7-6f60-4d4a-8ef6-
bab2699e6ab8   Bound   Success
```

Um backend foi criado com sucesso e vinculado ao `TridentBackendConfig` CR.

A fase pode ter um dos seguintes valores:

- `Bound`: O `TridentBackendConfig` CR está associado a um back-end, e esse backend contém `configRef` definido como `TridentBackendConfig` uid do CR.
- `Unbound`: Representado `""` usando . O `TridentBackendConfig` objeto não está vinculado a um backend. Todos os CRS recém-criados `TridentBackendConfig` estão nesta fase por padrão. Após as alterações de fase, ela não pode voltar a Unbound.
- `Deleting`: Os `TridentBackendConfig` CR's `deletionPolicy` foram definidos para eliminar. Quando

o `TridentBackendConfig` CR é excluído, ele passa para o estado de exclusão.

- Se não existirem declarações de volume persistentes (PVCs) no back-end, a exclusão do resultará na exclusão do `TridentBackendConfig` Trident do back-end, bem como do `TridentBackendConfig` CR.
- Se um ou mais PVCs estiverem presentes no back-end, ele vai para um estado de exclusão. Posteriormente, o `TridentBackendConfig` CR entra também na fase de eliminação. O back-end e `TridentBackendConfig` são excluídos somente depois que todos os PVCs são excluídos.

- `Lost`: O back-end associado ao `TridentBackendConfig` CR foi acidentalmente ou deliberadamente excluído e o `TridentBackendConfig` CR ainda tem uma referência ao back-end excluído. O `TridentBackendConfig` CR ainda pode ser eliminado independentemente do `deletionPolicy` valor.
- `Unknown`: O Trident não consegue determinar o estado ou a existência do back-end associado ao `TridentBackendConfig` CR. Por exemplo, se o servidor de API não estiver respondendo ou se o `tridentbackends.trident.netapp.io` CRD estiver ausente. Isso pode exigir intervenção.

Nesta fase, um backend é criado com sucesso! Existem várias operações que podem ser tratadas adicionalmente, "atualizações de back-end e exclusões de back-end"como o .

(Opcional) passo 4: Obtenha mais detalhes

Você pode executar o seguinte comando para obter mais informações sobre seu back-end:

```
kubectl -n trident get tbc backend-tbc-ontap-san -o wide
```

```
NAME                    BACKEND NAME        BACKEND UUID
PHASE   STATUS    STORAGE DRIVER   DELETION POLICY
backend-tbc-ontap-san   ontap-san-backend   8d24fce7-6f60-4d4a-8ef6-
bab2699e6ab8   Bound   Success   ontap-san        delete
```

Além disso, você também pode obter um despejo YAML/JSON do `TridentBackendConfig`.

```
kubectl -n trident get tbc backend-tbc-ontap-san -o yaml
```

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  creationTimestamp: 2021-04-21T20:45:11Z
  finalizers:
    - trident.netapp.io
  generation: 1
  name: backend-tbc-ontap-san
  namespace: trident
  resourceVersion: "947143"
  uid: 35b9d777-109f-43d5-8077-c74a4559d09c
```

```
spec:
  backendName: ontap-san-backend
  credentials:
    name: backend-tbc-ontap-san-secret
  managementLIF: 10.0.0.1
  dataLIF: 10.0.0.2
  storageDriverName: ontap-san
  svm: trident_svm
  version: 1
status:
  backendInfo:
    backendName: ontap-san-backend
    backendUUID: 8d24fce7-6f60-4d4a-8ef6-bab2699e6ab8
  deletionPolicy: delete
  lastOperationStatus: Success
  message: Backend 'ontap-san-backend' created
  phase: Bound
```

`backendInfo` Contém o `backendName` e o `backendUUID` do back-end criado em resposta ao `TridentBackendConfig` CR. O `lastOperationStatus` campo representa o status da última operação do `TridentBackendConfig` CR, que pode ser acionada pelo usuário (por exemplo, o usuário mudou algo no `spec`) ou acionado pelo Trident (por exemplo, durante reinicializações do Trident). Pode ser sucesso ou falhou. `phase` Representa o status da relação entre o `TridentBackendConfig` CR e o back-end. No exemplo acima, `phase` tem o valor vinculado, o que significa que o `TridentBackendConfig` CR está associado ao back-end.

Você pode executar o `kubectl -n trident describe tbc <tbc-cr-name>` comando para obter detalhes dos logs de eventos.

AVISO	Não é possível atualizar ou excluir um back-end que contenha um objeto `tridentctl` associado `TridentBackendConfig` usando o . Compreender as etapas envolvidas na troca entre `tridentctl` e `TridentBackendConfig`, "veja aqui".

Gerenciar backends

Gerencie os backends do Trident com kubectl

Saiba mais sobre como executar operações de gerenciamento de back-end usando `kubectl`o .

Excluir um back-end

Ao excluir um `TridentBackendConfig`, você instrui o Trident a excluir/reter backends (com base `deletionPolicy` no). Para excluir um back-end, certifique-se de que `deletionPolicy` está definido para excluir. Para eliminar apenas o `TridentBackendConfig`, certifique-se de que `deletionPolicy` está definido como reter. Isso garante que o backend ainda esteja presente e possa ser gerenciado usando `tridentctl`o .

Execute o seguinte comando:

```
kubectl delete tbc <tbc-name> -n trident
```

O Trident não exclui os segredos do Kubernetes que estavam em uso `TridentBackendConfig` pelo . O usuário do Kubernetes é responsável pela limpeza de segredos. Cuidado deve ser tomado ao excluir segredos. Você deve excluir segredos somente se eles não estiverem em uso pelos backends.

Veja os backends existentes

Execute o seguinte comando:

```
kubectl get tbc -n trident
```

Você também pode executar `tridentctl get backend -n trident` ou `tridentctl get backend -o yaml -n trident` obter uma lista de todos os backends que existem. Esta lista também incluirá backends que foram criados com `tridentctl`.

Atualize um back-end

Pode haver várias razões para atualizar um backend:

- As credenciais para o sistema de storage foram alteradas. Para atualizar as credenciais, o segredo do Kubernetes que é usado no `TridentBackendConfig` objeto deve ser atualizado. O Trident atualizará automaticamente o back-end com as credenciais mais recentes fornecidas. Execute o seguinte comando para atualizar o segredo do Kubernetes:

  ```
  kubectl apply -f <updated-secret-file.yaml> -n trident
  ```

- Os parâmetros (como o nome do SVM do ONTAP sendo usado) precisam ser atualizados.
  - Você pode atualizar `TridentBackendConfig` objetos diretamente pelo Kubernetes usando o seguinte comando:

    ```
    kubectl apply -f <updated-backend-file.yaml>
    ```

  - Alternativamente, você pode fazer alterações no CR existente `TridentBackendConfig` usando o seguinte comando:

    ```
    kubectl edit tbc <tbc-name> -n trident
    ```

OBSERVAÇÃO

- Se uma atualização de back-end falhar, o back-end continuará em sua última configuração conhecida. Pode visualizar os registos para determinar a causa executando `kubectl get tbc <tbc-name> -o yaml -n trident` ou `kubectl describe tbc <tbc-name> -n trident`.
- Depois de identificar e corrigir o problema com o arquivo de configuração, você pode executar novamente o comando update.

Gerencie os backends do Trident com tridentctl

Saiba mais sobre como executar operações de gerenciamento de back-end usando `tridentctl`o .

Crie um backend

Depois de criar um "arquivo de configuração de back-end", execute o seguinte comando:

```
tridentctl create backend -f <backend-file> -n trident
```

Se a criação do backend falhar, algo estava errado com a configuração do backend. Você pode exibir os logs para determinar a causa executando o seguinte comando:

```
tridentctl logs -n trident
```

Depois de identificar e corrigir o problema com o arquivo de configuração, você pode simplesmente executar o `create` comando novamente.

Excluir um back-end

Para excluir um back-end do Trident, faça o seguinte:

1. Recuperar o nome do backend:

   ```
   tridentctl get backend -n trident
   ```

2. Excluir o backend:

   ```
   tridentctl delete backend <backend-name> -n trident
   ```

OBSERVAÇÃO	Se o Trident provisionou volumes e snapshots desse back-end que ainda existem, excluir o back-end impede que novos volumes sejam provisionados por ele. O backend continuará a existir em um estado de "exclusão".

Veja os backends existentes

Para visualizar os backends que o Trident conhece, faça o seguinte:

- Para obter um resumo, execute o seguinte comando:

  ```
  tridentctl get backend -n trident
  ```

- Para obter todos os detalhes, execute o seguinte comando:

```
tridentctl get backend -o json -n trident
```

Atualize um back-end

Depois de criar um novo arquivo de configuração de back-end, execute o seguinte comando:

```
tridentctl update backend <backend-name> -f <backend-file> -n trident
```

Se a atualização do backend falhar, algo estava errado com a configuração do backend ou você tentou uma atualização inválida. Você pode exibir os logs para determinar a causa executando o seguinte comando:

```
tridentctl logs -n trident
```

Depois de identificar e corrigir o problema com o arquivo de configuração, você pode simplesmente executar o `update` comando novamente.

Identificar as classes de armazenamento que usam um back-end

Este é um exemplo do tipo de perguntas que você pode responder com o JSON que `tridentctl` produz para objetos de back-end. Isso usa o `jq` utilitário, que você precisa instalar.

```
tridentctl get backend -o json | jq '[.items[] | {backend: .name,
storageClasses: [.storage[].storageClasses]|unique}]'
```

Isso também se aplica a backends que foram criados usando `TridentBackendConfig`o .

Escolha um método de gerenciamento de backend do Trident

Saiba mais sobre as diferentes maneiras de gerenciar backends no Trident.

Opções para gerenciar backends

Com a introdução `TridentBackendConfig`do , os administradores agora têm duas maneiras exclusivas de gerenciar backends. Isso coloca as seguintes perguntas:

- Os backends podem ser criados usando `tridentctl` ser gerenciados com `TridentBackendConfig`?
- Os backends podem ser criados usando `TridentBackendConfig` ser gerenciados `tridentctl` usando ?

Gerenciar `tridentctl` backends usando `TridentBackendConfig`

Esta seção aborda as etapas necessárias para gerenciar backends que foram criados usando `tridentctl` diretamente a interface do Kubernetes criando `TridentBackendConfig` objetos.

Isso se aplicará aos seguintes cenários:

- Backends pré-existentes, que não têm um `TridentBackendConfig` porque foram criados com `tridentctl`.
- Novos backends que foram criados com `tridentctl`, enquanto outros `TridentBackendConfig` objetos existem.

Em ambos os cenários, os backends continuarão a estar presentes, com o Trident agendando volumes e operando neles. Os administradores têm uma das duas opções aqui:

- Continue `tridentctl` usando para gerenciar backends que foram criados usando-o.
- Vincular backends criados usando `tridentctl` a um novo `TridentBackendConfig` objeto. Fazer isso significaria que os backends serão gerenciados usando `kubectl` e não `tridentctl`.

Para gerenciar um back-end pré-existente usando `kubectl`o , você precisará criar um `TridentBackendConfig` que se vincule ao back-end existente. Aqui está uma visão geral de como isso funciona:

1. Crie um segredo do Kubernetes. O segredo contém as credenciais que a Trident precisa para se comunicar com o cluster/serviço de storage.
2. Crie `TridentBackendConfig` um objeto. Isso contém detalhes sobre o cluster/serviço de armazenamento e faz referência ao segredo criado na etapa anterior. Deve-se ter cuidado para especificar parâmetros de configuração idênticos (como `spec.backendName` , , `spec.storagePrefix`, `spec.storageDriverName` e assim por diante). `spec.backendName` deve ser definido como o nome do backend existente.

Passo 0: Identifique o backend

Para criar um `TridentBackendConfig` que se vincula a um backend existente, você precisará obter a configuração de backend. Neste exemplo, vamos supor que um backend foi criado usando a seguinte definição JSON:

```
tridentctl get backend ontap-nas-backend -n trident
+---------------------+----------------
+--------------------------------------+--------+---------+
|          NAME       | STORAGE DRIVER |                 UUID
| STATE   | VOLUMES |
+---------------------+----------------
+--------------------------------------+--------+---------+
| ontap-nas-backend   | ontap-nas      | 52f2eb10-e4c6-4160-99fc-
96b3be5ab5d7 | online |      25 |
+---------------------+----------------
+--------------------------------------+--------+---------+
```

```
cat ontap-nas-backend.json
```

```
{
  "version": 1,
```

```
  "storageDriverName": "ontap-nas",
  "managementLIF": "10.10.10.1",
  "dataLIF": "10.10.10.2",
  "backendName": "ontap-nas-backend",
  "svm": "trident_svm",
  "username": "cluster-admin",
  "password": "admin-password",
  "defaults": {
    "spaceReserve": "none",
    "encryption": "false"
  },
  "labels": {
    "store": "nas_store"
  },
  "region": "us_east_1",
  "storage": [
    {
      "labels": {
        "app": "msoffice",
        "cost": "100"
      },
      "zone": "us_east_1a",
      "defaults": {
        "spaceReserve": "volume",
        "encryption": "true",
        "unixPermissions": "0755"
      }
    },
    {
      "labels": {
        "app": "mysqldb",
        "cost": "25"
      },
      "zone": "us_east_1d",
      "defaults": {
        "spaceReserve": "volume",
        "encryption": "false",
        "unixPermissions": "0775"
      }
    }
  ]
}
```

Etapa 1: Crie um segredo do Kubernetes

Crie um segredo que contenha as credenciais para o back-end, como mostrado neste exemplo:

```
cat tbc-ontap-nas-backend-secret.yaml
```

```
apiVersion: v1
kind: Secret
metadata:
  name: ontap-nas-backend-secret
type: Opaque
stringData:
  username: cluster-admin
  password: admin-password
```

```
kubectl create -f tbc-ontap-nas-backend-secret.yaml -n trident
secret/backend-tbc-ontap-san-secret created
```

Passo 2: Crie um `TridentBackendConfig` CR

O próximo passo é criar um `TridentBackendConfig` CR que se vinculará automaticamente ao pré-existente `ontap-nas-backend` (como neste exemplo). Certifique-se de que os seguintes requisitos são cumpridos:

- O mesmo nome de back-end é definido no `spec.backendName`.
- Os parâmetros de configuração são idênticos ao back-end original.
- Os pools virtuais (se presentes) devem manter a mesma ordem que no back-end original.
- As credenciais são fornecidas por meio de um segredo do Kubernetes e não em texto simples.

Neste caso, o `TridentBackendConfig` será parecido com este:

```
cat backend-tbc-ontap-nas.yaml
```

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: tbc-ontap-nas-backend
spec:
  version: 1
  storageDriverName: ontap-nas
  managementLIF: 10.10.10.1
  dataLIF: 10.10.10.2
  backendName: ontap-nas-backend
  svm: trident_svm
  credentials:
```

```
    name: mysecret
  defaults:
    spaceReserve: none
    encryption: 'false'
  labels:
    store: nas_store
  region: us_east_1
  storage:
  - labels:
      app: msoffice
      cost: '100'
    zone: us_east_1a
    defaults:
      spaceReserve: volume
      encryption: 'true'
      unixPermissions: '0755'
  - labels:
      app: mysqldb
      cost: '25'
    zone: us_east_1d
    defaults:
      spaceReserve: volume
      encryption: 'false'
      unixPermissions: '0775'
```

```
kubectl create -f backend-tbc-ontap-nas.yaml -n trident
tridentbackendconfig.trident.netapp.io/tbc-ontap-nas-backend created
```

Etapa 3: Verifique o status do `TridentBackendConfig` CR

Após a criação do `TridentBackendConfig` , sua fase deve ser `Bound`. Ele também deve refletir o mesmo nome de back-end e UUID que o do back-end existente.

```
kubectl get tbc tbc-ontap-nas-backend -n trident
NAME                  BACKEND NAME          BACKEND UUID
PHASE   STATUS
tbc-ontap-nas-backend  ontap-nas-backend     52f2eb10-e4c6-4160-99fc-
96b3be5ab5d7   Bound   Success

#confirm that no new backends were created (i.e., TridentBackendConfig did
not end up creating a new backend)
tridentctl get backend -n trident
+--------------------+---------------
+--------------------------------------+---------+---------+
```

```
|          NAME       | STORAGE DRIVER |                 UUID
| STATE  | VOLUMES |
+---------------------+----------------
+--------------------------------------+--------+---------+
| ontap-nas-backend   | ontap-nas      | 52f2eb10-e4c6-4160-99fc-
96b3be5ab5d7 | online |      25 |
+---------------------+----------------
+--------------------------------------+--------+---------+
```

O backend agora será completamente gerenciado usando o `tbc-ontap-nas-backend TridentBackendConfig` objeto.

Gerenciar `TridentBackendConfig` backends usando `tridentctl`

```
`tridentctl` pode ser usado para listar backends que foram criados usando
`TridentBackendConfig`. Além disso, os administradores também podem optar
por gerenciar completamente esses backends `tridentctl` excluindo
`TridentBackendConfig` e certificando-se de `spec.deletionPolicy` que está
definido como `retain`.
```

Passo 0: Identifique o backend

Por exemplo, vamos supor que o seguinte backend foi criado usando `TridentBackendConfig`:

```
kubectl get tbc backend-tbc-ontap-san -n trident -o wide
NAME                    BACKEND NAME        BACKEND UUID
PHASE   STATUS    STORAGE DRIVER   DELETION POLICY
backend-tbc-ontap-san   ontap-san-backend   81abcb27-ea63-49bb-b606-
0a5315ac5f82   Bound   Success   ontap-san        delete

tridentctl get backend ontap-san-backend -n trident
+-------------------+----------------
+--------------------------------------+--------+---------+
|       NAME        | STORAGE DRIVER |                 UUID
| STATE  | VOLUMES |
+-------------------+----------------
+--------------------------------------+--------+---------+
| ontap-san-backend | ontap-san      | 81abcb27-ea63-49bb-b606-
0a5315ac5f82 | online |      33 |
+-------------------+----------------
+--------------------------------------+--------+---------+
```

A partir da saída, vê-se que `TridentBackendConfig` foi criado com sucesso e está vinculado a um backend [observe o UUID do backend].

Passo 1: Confirmar `deletionPolicy` está definido como `retain`

Vamos dar uma olhada no valor `deletionPolicy` de . Isso precisa ser definido como `retain`. Isso garante que quando um `TridentBackendConfig` CR é excluído, a definição de back-end ainda estará presente e pode ser gerenciada com `tridentctl`o .

```
kubectl get tbc backend-tbc-ontap-san -n trident -o wide
NAME                    BACKEND NAME        BACKEND UUID
PHASE   STATUS    STORAGE DRIVER   DELETION POLICY
backend-tbc-ontap-san   ontap-san-backend   81abcb27-ea63-49bb-b606-
0a5315ac5f82   Bound   Success   ontap-san        delete

# Patch value of deletionPolicy to retain
kubectl patch tbc backend-tbc-ontap-san --type=merge -p
'{"spec":{"deletionPolicy":"retain"}}' -n trident
tridentbackendconfig.trident.netapp.io/backend-tbc-ontap-san patched

#Confirm the value of deletionPolicy
kubectl get tbc backend-tbc-ontap-san -n trident -o wide
NAME                    BACKEND NAME        BACKEND UUID
PHASE   STATUS    STORAGE DRIVER   DELETION POLICY
backend-tbc-ontap-san   ontap-san-backend   81abcb27-ea63-49bb-b606-
0a5315ac5f82   Bound   Success   ontap-san        retain
```

OBSERVAÇÃO	Não avance para o passo seguinte, a menos `deletionPolicy` que esteja definido para `retain`.

Etapa 2: Exclua o `TridentBackendConfig` CR

O passo final é eliminar o `TridentBackendConfig` CR. Depois de confirmar que o `deletionPolicy` está definido como `retain`, pode avançar com a eliminação:

```
kubectl delete tbc backend-tbc-ontap-san -n trident
tridentbackendconfig.trident.netapp.io "backend-tbc-ontap-san" deleted

tridentctl get backend ontap-san-backend -n trident
+------------------+---------------
+--------------------------------------+--------+---------+
|       NAME        | STORAGE DRIVER |                 UUID
| STATE  | VOLUMES |
+------------------+---------------
+--------------------------------------+--------+---------+
| ontap-san-backend | ontap-san      | 81abcb27-ea63-49bb-b606-
0a5315ac5f82 | online |      33 |
+------------------+---------------
```

```
+--------------------------------------+--------+---------+
```

Após a exclusão `TridentBackendConfig` do objeto, o Trident simplesmente o remove sem realmente excluir o próprio backend.

Criar e gerenciar classes de armazenamento

Crie uma classe de armazenamento do Trident

Configure um objeto Kubernetes StorageClass e crie a classe de storage para instruir o Trident a provisionar volumes.

Configurar um objeto Kubernetes StorageClass

O "Objeto Kubernetes StorageClass" identifica o Trident como o provisionador usado para essa classe e instrui o Trident a provisionar um volume. Por exemplo:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-gold
provisioner: csi.trident.netapp.io
mountOptions:
  - nfsvers=3
  - nolock
parameters:
  backendType: "ontap-nas"
  media: "ssd"
allowVolumeExpansion: true
volumeBindingMode: Immediate
```

"Objetos Kubernetes e Trident"Consulte para obter detalhes sobre como as classes de armazenamento interagem com os `PersistentVolumeClaim` parâmetros e para controlar como o Trident provisiona volumes.

Crie uma classe de armazenamento

Depois de criar o objeto StorageClass, você pode criar a classe de armazenamento. Amostras da classe de armazenamento fornece algumas amostras básicas que você pode usar ou modificar.

Passos

1. Esse é um objeto do Kubernetes, então use `kubectl` para criá-lo no Kubernetes.

   ```
   kubectl create -f sample-input/storage-class-basic-csi.yaml
   ```

2. Agora você deve ver uma classe de armazenamento **Basic-csi** no Kubernetes e no Trident, e o Trident

deve ter descoberto os pools no back-end.

```
kubectl get sc basic-csi
```

```
NAME         PROVISIONER             AGE
basic-csi    csi.trident.netapp.io   15h
```

```
./tridentctl -n trident get storageclass basic-csi -o json
```

```
{
  "items": [
    {
      "Config": {
        "version": "1",
        "name": "basic-csi",
        "attributes": {
          "backendType": "ontap-nas"
        },
        "storagePools": null,
        "additionalStoragePools": null
      },
      "storage": {
        "ontapnas_10.0.0.1": [
          "aggr1",
          "aggr2",
          "aggr3",
          "aggr4"
        ]
      }
    }
  ]
}
```

Amostras da classe de armazenamento

O Trident fornece "definições de classe de armazenamento simples para backends específicos".

Alternativamente, você pode editar `sample-input/storage-class-csi.yaml.templ` o arquivo que vem com o instalador e substituir *`BACKEND_TYPE`* pelo nome do driver de armazenamento.

```
./tridentctl -n trident get backend
+-------------+----------------+--------------------------------------
```

```
+--------+---------+
|    NAME     | STORAGE DRIVER |                 UUID                 |
STATE  | VOLUMES |
+-------------+---------------+--------------------------------------
+--------+---------+
| nas-backend | ontap-nas      | 98e19b74-aec7-4a3d-8dcf-128e5033b214 |
online |       0 |
+-------------+---------------+--------------------------------------
+--------+---------+

cp sample-input/storage-class-csi.yaml.templ sample-input/storage-class-
basic-csi.yaml

# Modify __BACKEND_TYPE__ with the storage driver field above (e.g.,
ontap-nas)
vi sample-input/storage-class-basic-csi.yaml
```

Gerencie as classes de armazenamento do Trident

Você pode exibir classes de armazenamento existentes, definir uma classe de armazenamento padrão, identificar o back-end da classe de armazenamento e excluir classes de armazenamento.

Exibir as classes de armazenamento existentes

- Para visualizar as classes de armazenamento do Kubernetes existentes, execute o seguinte comando:

```
kubectl get storageclass
```

- Para ver os detalhes da classe de storage do Kubernetes, execute o seguinte comando:

```
kubectl get storageclass <storage-class> -o json
```

- Para exibir as classes de armazenamento sincronizadas do Trident, execute o seguinte comando:

```
tridentctl get storageclass
```

- Para exibir os detalhes da classe de armazenamento sincronizado do Trident, execute o seguinte comando:

```
tridentctl get storageclass <storage-class> -o json
```

Defina uma classe de armazenamento padrão

O Kubernetes 1,6 adicionou a capacidade de definir uma classe de storage padrão. Esta é a classe de armazenamento que será usada para provisionar um volume persistente se um usuário não especificar um em uma reivindicação de volume persistente (PVC).

- Defina uma classe de armazenamento padrão definindo a anotação `storageclass.kubernetes.io/is-default-class` como verdadeira na definição da classe de armazenamento. De acordo com a especificação, qualquer outro valor ou ausência da anotação é interpretado como falso.
- Você pode configurar uma classe de armazenamento existente para ser a classe de armazenamento padrão usando o seguinte comando:

```
kubectl patch storageclass <storage-class-name> -p '{"metadata":
{"annotations":{"storageclass.kubernetes.io/is-default-class":"true"}}}'
```

- Da mesma forma, você pode remover a anotação de classe de armazenamento padrão usando o seguinte comando:

```
kubectl patch storageclass <storage-class-name> -p '{"metadata":
{"annotations":{"storageclass.kubernetes.io/is-default-class":"false"}}}'
```

Há também exemplos no pacote de instalação do Trident que incluem esta anotação.

OBSERVAÇÃO	Deve haver apenas uma classe de armazenamento padrão no cluster de cada vez. O Kubernetes não impede tecnicamente que você tenha mais de um, mas se comportará como se não houvesse nenhuma classe de storage padrão.

Identificar o back-end de uma classe de storage

Este é um exemplo do tipo de perguntas que você pode responder com o JSON que `tridentctl` produz para objetos backend do Trident. Isso usa o `jq` utilitário, que você pode precisar instalar primeiro.

```
tridentctl get storageclass -o json | jq  '[.items[] | {storageClass:
.Config.name, backends: [.storage]|unique}]'
```

Excluir uma classe de armazenamento

Para excluir uma classe de armazenamento do Kubernetes, execute o seguinte comando:

```
kubectl delete storageclass <storage-class>
```

`<storage-class>` deve ser substituído pela sua classe de armazenamento.

Todos os volumes persistentes criados com essa classe de storage permanecerão intocados, e o Trident continuará gerenciando-os.

OBSERVAÇÃO	O Trident impõe um espaço em branco `fsType` para os volumes que cria. Para backends iSCSI, recomenda-se aplicar `parameters.fsType` no StorageClass. Você deve excluir StorageClasses existentes e recriá-los com `parameters.fsType` especificado.

Provisionar e gerenciar volumes

Provisione um volume do Trident

Crie um PersistentVolumeClaim (PVC) que use o Kubernetes StorageClass configurado para solicitar acesso ao PV. Em seguida, pode montar o PV num pod.

Visão geral

Um "*PersistentVolumeClaim*" (PVC) é um pedido de acesso ao Persistentvolume no cluster.

O PVC pode ser configurado para solicitar o armazenamento de um determinado tamanho ou modo de acesso. Usando o StorageClass associado, o administrador do cluster pode controlar mais do que o Persistentvolume e o modo de acesso, como desempenho ou nível de serviço.

Depois de criar o PVC, você pode montar o volume em um pod.

Crie o PVC

Passos

1. Crie o PVC.

```
kubectl create -f pvc.yaml
```

2. Verifique o estado do PVC.

```
kubectl get pvc
```

```
NAME          STATUS   VOLUME     CAPACITY   ACCESS MODES   STORAGECLASS AGE
pvc-storage   Bound    pv-name    1Gi        RWO                    5m
```

1. Monte o volume num pod.

```
kubectl create -f pv-pod.yaml
```

OBSERVAÇÃO	Pode monitorizar o progresso utilizando `kubectl get pod --watch`o .

2. Verifique se o volume está montado no `/my/mount/path`.

```
kubectl exec -it task-pv-pod -- df -h /my/mount/path
```

3. Agora você pode excluir o Pod. O aplicativo Pod não existirá mais, mas o volume permanecerá.

```
kubectl delete pod pv-pod
```

Manifestos de amostra

▼ **PersistentVolumeClaim amostra manifestos**

Estes exemplos mostram opções básicas de configuração de PVC.

PVC com acesso RWO

Este exemplo mostra um PVC básico com acesso RWO associado a um StorageClass `basic-csi` chamado .

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc-storage
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: basic-csi
```

PVC com NVMe/TCP

Este exemplo mostra um PVC básico para NVMe/TCP com acesso RWO associado a um StorageClass `protection-gold` chamado .

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
name: pvc-san-nvme
spec:
accessModes:
  - ReadWriteOnce
resources:
  requests:
    storage: 300Mi
storageClassName: protection-gold
```

▼ Amostras de manifesto POD

Estes exemplos mostram configurações básicas para anexar o PVC a um pod.

Configuração básica

```
kind: Pod
apiVersion: v1
metadata:
  name: pv-pod
spec:
  volumes:
    - name: storage
      persistentVolumeClaim:
       claimName: pvc-storage
  containers:
    - name: pv-container
      image: nginx
      ports:
        - containerPort: 80
          name: "http-server"
      volumeMounts:
        - mountPath: "/my/mount/path"
          name: storage
```

Configuração básica NVMe/TCP

```
apiVersion: v1
kind: Pod
metadata:
  name: pod-nginx
spec:
  volumes:
    - name: basic-pvc
      persistentVolumeClaim:
        claimName: pvc-san-nvme
  containers:
    - name: task-pv-container
      image: nginx
      volumeMounts:
        - mountPath: "/my/mount/path"
          name: basic-pvc
```

"Objetos Kubernetes e Trident"Consulte para obter detalhes sobre como as classes de armazenamento interagem com os `PersistentVolumeClaim` parâmetros e para controlar como o Trident provisiona volumes.

Mover um volume

A partir do Trident 26.06, um administrador de storage pode mover de forma não disruptiva um volume iSCSI para um nó ONTAP diferente e agregá-lo dentro do mesmo cluster ONTAP enquanto a carga de trabalho que o utiliza continua em execução. Você solicita a movimentação de forma declarativa com um `TridentVolumeMove` recurso personalizado ou o `tridentctl` equivalente, e Trident coordena ONTAP e o host para que a E/S nunca seja interrompida.

Requisitos

A movimentação de volume aplica-se ao driver `ontap-san` para volumes iSCSI apenas. Os protocolos NVMe e NAS não são suportados.

Antes de mover um volume, certifique-se de que os seguintes requisitos sejam atendidos:

- O backend utiliza o `ontap-san` driver com a API REST do ONTAP.
- O cluster ONTAP possui pelo menos dois nós.
- Você possui um dos seguintes conjuntos de credenciais:
  - ONTAP 9.19.1 ou posterior se você estiver usando a função ONTAP personalizada dedicada.
  - Credenciais de administrador de cluster em versões anteriores do ONTAP.

Como funciona

Trident move o volume (LUN) entre dois locais dentro do mesmo cluster ONTAP. Antes da movimentação dos dados, Trident adiciona novos caminhos iSCSI ao destino, juntamente com os caminhos existentes, para que o dispositivo multipath do host nunca perca o acesso. Após a conclusão da movimentação pelo ONTAP, Trident remove os caminhos de origem obsoletos.

Trident executa a movimentação por meio de um conjunto ordenado de estados que são visíveis no `TridentVolumeMove` recurso, para que você possa acompanhar o progresso da movimentação.

Mover um volume

Identifique o volume e os locais

1. Encontre o nome do PV que deseja mover. O nome do PV também é o nome do recurso de movimentação.
2. No cluster ONTAP, anote o nó e o agregado atuais (de origem) para esse volume e escolha o nó de destino e o agregado de destino (alvo).

Inicie a movimentação

Use qualquer um dos métodos. Ambos criam o mesmo `TridentVolumeMove`recurso, cujo nome é o nome da PV.

Usando tridentctl

```
tridentctl create volume-move \
```

```
  --volume      pvc-8a814d62-bd58-4253-b0d1-82f2885db671 \
  --source-node ontap-node-01 \
  --source-pool aggr1_node01 \
  --target-node ontap-node-02 \
  --target-pool aggr1_node02 \
  -n trident
```

`tvm` é um alias para `volume-move`. Adicione `--delete-after-success 10m` para remover o recurso automaticamente após a movimentação ser concluída com sucesso.

Utilizando um recurso TridentVolumeMove

```
apiVersion: trident.netapp.io/v1
kind: TridentVolumeMove
metadata:
  name: pvc-8a814d62-bd58-4253-b0d1-82f2885db671   # the PV to move
  namespace: trident
spec:
  sourceNode: ontap-node-01
  sourcePool: aggr1_node01
  targetNode: ontap-node-02
  targetPool: aggr1_node02
  # deleteAfterSuccess: 10m   # optional
```

Aplique o recurso:

```
kubectl apply -f tridentvolumemove.yaml
```

Campos de especificação

`sourceNode`/ `targetNode`

Nomes de nós ONTAP (não UUIDs).

`sourcePool`/ `targetPool`

Nomes de agregados do ONTAP.

`deleteAfterSuccess`

Duração opcional, como `30s` ou `10m`. Defina como `0` para excluir o recurso imediatamente ou omita para manter o recurso.

Monitore a movimentação

Acompanhe o progresso usando o `tvm`nome abreviado ou `tridentvolumemove`:

```
kubectl get tridentvolumemove -n trident
```

Para visualizar o estado atual e o progresso por nó, descreva o recurso:

```
kubectl describe tridentvolumemove <pv-name> -n trident
```

O volume pode ter anexos simultâneos durante o `Moving` estado, porque ONTAP pode levar um tempo considerável para concluir a movimentação de um volume grande. Para garantir uma movimentação perfeita, Trident coordena a comunicação entre o Trident controller e o Trident node. Cada estado executa as seguintes etapas em ordem, terminando em `Succeeded` ou `Failed`.

Estado	O que está acontecendo
`Pending`	Trident aceita e valida a solicitação de movimentação e reúne as informações necessárias antes que qualquer alteração no storage seja iniciada.
`ControllerStaging`	Trident prepara o local de destino. Ele mapeia as interfaces de armazenamento do nó de destino para o LUN e atualiza o Kubernetes para que uma futura reinicialização se reconecte ao local correto.
`NodeStaging`	Em cada host que possui o volume montado, Trident adiciona os novos caminhos de storage do nó de destino juntamente com os caminhos existentes, de forma que a E/S do pod continue em ambos os conjuntos de caminhos.
`Moving`	ONTAP copia e transfere os dados do volume para o agregado de destino. Esta é a fase mais longa, e ONTAP a gerencia completamente enquanto Trident faz polling até terminar. Novas publicações são permitidas durante este estado.
`ControllerUnstaging`	A movimentação de dados foi concluída. Trident atualiza o Kubernetes para remover os caminhos de origem do estado desejado e, em seguida, desmapeia as interfaces de storage do nó de origem do LUN.
`NodeUnstaging`	Em cada host, Trident remove os caminhos de origem obsoletos. O host fica apenas com os caminhos de destino, e o dispositivo multipath continua a fornecer E/S sem interrupção.
`Succeeded`	O volume foi totalmente movido e está online no destino. Trident retém o `TridentVolumeMove` recurso ou o exclui automaticamente se você definir `deleteAfterSuccess`.
`Failed`	Um erro irrecuperável interrompeu a movimentação ou impediu seu início. Trident redefine o estado do volume para que as operações normais possam ser retomadas e mantém o recurso com uma mensagem de erro. Analise a mensagem, corrija a causa raiz, exclua o recurso e tente novamente.

A `.status.attachments`lista indica o progresso de cada nó conectado em seu campo `state`:

Estado de anexação	Significado
`Pending`	Nó identificado como conectado; o trabalho ainda não começou.

Estado de anexação	Significado
`Bridged`	Novos caminhos de destino adicionados juntamente com os caminhos existentes.
`Migrated`	O tráfego agora é atendido pelos novos caminhos.
`Cleaned`	Caminhos de origem obsoletos removidos do nó.
`Detached`	Nó desprendido durante a movimentação e excluído de análises posteriores enquanto Trident avalia se o anexo está pronto para prosseguir para a próxima etapa.
`Failed`	Erro terminal neste nó. Veja a mensagem.

Limpe o recurso de movimentação

Se você definir `deleteAfterSuccess`, Trident remove o recurso automaticamente. Caso contrário, você pode excluí-lo manualmente. Antes de solicitar outra movimentação no mesmo volume, certifique-se de que não exista nenhum recurso `TridentVolumeMove` para esse volume em um estado `Succeeded` ou `Failed`.

OBSERVAÇÃO	Trident nunca exclui automaticamente as movimentações com falha. Trident mantém um `Failed` recurso como registro de auditoria. Depois de corrigir a causa raiz, exclua o recurso manualmente antes de tentar novamente.

Expandir volumes do Trident

O Trident oferece aos usuários do Kubernetes a capacidade de expandir seus volumes depois que eles são criados. Encontre informações sobre as configurações necessárias para expandir volumes iSCSI, NFS, SMB, NVMe/TCP e FC.

Expanda um volume iSCSI

É possível expandir um iSCSI Persistent volume (PV) usando o provisionador de CSI.

OBSERVAÇÃO	A expansão de volume iSCSI é suportada pelos `ontap-san` `ontap-san-economy` drivers , , `solidfire-san` e requer o Kubernetes 1,16 e posterior.

Etapa 1: Configure o StorageClass para dar suporte à expansão de volume

Edite a definição StorageClass para definir o `allowVolumeExpansion` campo como `true`.

```
cat storageclass-ontapsan.yaml
```

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-san
provisioner: csi.trident.netapp.io
parameters:
```

```
  backendType: "ontap-san"
allowVolumeExpansion: True
```

Para um StorageClass já existente, edite-o para incluir o `allowVolumeExpansion` parâmetro.

Etapa 2: Crie um PVC com o StorageClass que você criou

Edite a definição de PVC e atualize o `spec.resources.requests.storage` para refletir o tamanho recém-desejado, que deve ser maior do que o tamanho original.

```
cat pvc-ontapsan.yaml
```

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: san-pvc
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: ontap-san
```

O Trident cria um volume persistente (PV) e associa-o a esta reivindicação de volume persistente (PVC).

```
kubectl get pvc
NAME      STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS   AGE
san-pvc   Bound    pvc-8a814d62-bd58-4253-b0d1-82f2885db671   1Gi
RWO            ontap-san      8s

kubectl get pv
NAME                                       CAPACITY   ACCESS MODES
RECLAIM POLICY   STATUS   CLAIM              STORAGECLASS   REASON   AGE
pvc-8a814d62-bd58-4253-b0d1-82f2885db671   1Gi        RWO
Delete           Bound    default/san-pvc    ontap-san               10s
```

Passo 3: Defina um pod que prende o PVC

Fixe o PV a um pod para que ele seja redimensionado. Existem dois cenários ao redimensionar um iSCSI PV:

- Se o PV estiver conetado a um pod, o Trident expande o volume no back-end de armazenamento, refaz o dispositivo e redimensiona o sistema de arquivos.

- Ao tentar redimensionar um PV não anexado, o Trident expande o volume no back-end de armazenamento. Depois que o PVC é ligado a um pod, o Trident refaz o dispositivo e redimensiona o sistema de arquivos. Em seguida, o Kubernetes atualiza o tamanho do PVC após a operação de expansão ter sido concluída com sucesso.

Neste exemplo, é criado um pod que usa o `san-pvc`.

```
 kubectl get pod
NAME         READY   STATUS    RESTARTS   AGE
ubuntu-pod   1/1     Running   0          65s

 kubectl describe pvc san-pvc
Name:          san-pvc
Namespace:     default
StorageClass:  ontap-san
Status:        Bound
Volume:        pvc-8a814d62-bd58-4253-b0d1-82f2885db671
Labels:        <none>
Annotations:   pv.kubernetes.io/bind-completed: yes
               pv.kubernetes.io/bound-by-controller: yes
               volume.beta.kubernetes.io/storage-provisioner:
csi.trident.netapp.io
Finalizers:    [kubernetes.io/pvc-protection]
Capacity:      1Gi
Access Modes:  RWO
VolumeMode:    Filesystem
Mounted By:    ubuntu-pod
```

Passo 4: Expanda o PV

Para redimensionar o PV que foi criado de 1Gi a 2Gi, edite a definição de PVC e atualize o `spec.resources.requests.storage` para 2Gi.

```
kubectl edit pvc san-pvc
```

```
# Please edit the object below. Lines beginning with a '#' will be
ignored,
# and an empty file will abort the edit. If an error occurs while saving
this file will be
# reopened with the relevant failures.
#
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
```

```
    pv.kubernetes.io/bind-completed: "yes"
    pv.kubernetes.io/bound-by-controller: "yes"
    volume.beta.kubernetes.io/storage-provisioner: csi.trident.netapp.io
  creationTimestamp: "2019-10-10T17:32:29Z"
  finalizers:
  - kubernetes.io/pvc-protection
  name: san-pvc
  namespace: default
  resourceVersion: "16609"
  selfLink: /api/v1/namespaces/default/persistentvolumeclaims/san-pvc
  uid: 8a814d62-bd58-4253-b0d1-82f2885db671
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 2Gi
 # ...
```

Etapa 5: Validar a expansão

Você pode validar a expansão trabalhada corretamente verificando o tamanho do PVC, PV e o volume Trident:

```
kubectl get pvc san-pvc
NAME      STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS   AGE
san-pvc   Bound    pvc-8a814d62-bd58-4253-b0d1-82f2885db671   2Gi
RWO            ontap-san      11m
kubectl get pv
NAME                                       CAPACITY   ACCESS MODES
RECLAIM POLICY   STATUS   CLAIM              STORAGECLASS   REASON   AGE
pvc-8a814d62-bd58-4253-b0d1-82f2885db671   2Gi        RWO
Delete           Bound    default/san-pvc    ontap-san               12m
tridentctl get volumes -n trident
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+---------+---------+
|                  NAME                   |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID             | STATE   | MANAGED |
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+---------+---------+
| pvc-8a814d62-bd58-4253-b0d1-82f2885db671 | 2.0 GiB | ontap-san     |
block    | a9b7bfff-0505-4e31-b6c5-59f492e02d33 | online | true    |
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+---------+---------+
```

Expandir um volume FC

É possível expandir um volume persistente (PV) FC com o provisionador de CSI.

OBSERVAÇÃO	A expansão de volume de FC é compatível com `ontap-san` o driver e requer o Kubernetes 1,16 e posterior.

Etapa 1: Configure o StorageClass para dar suporte à expansão de volume

Edite a definição StorageClass para definir o `allowVolumeExpansion` campo como `true`.

```
cat storageclass-ontapsan.yaml
```

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-san
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
allowVolumeExpansion: True
```

Para um StorageClass já existente, edite-o para incluir o `allowVolumeExpansion` parâmetro.

Etapa 2: Crie um PVC com o StorageClass que você criou

Edite a definição de PVC e atualize o `spec.resources.requests.storage` para refletir o tamanho recém-desejado, que deve ser maior do que o tamanho original.

```
cat pvc-ontapsan.yaml
```

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: san-pvc
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: ontap-san
```

O Trident cria um volume persistente (PV) e associa-o a esta reivindicação de volume persistente (PVC).

```
kubectl get pvc
NAME      STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS   AGE
san-pvc   Bound    pvc-8a814d62-bd58-4253-b0d1-82f2885db671   1Gi
RWO            ontap-san      8s

kubectl get pv
NAME                                       CAPACITY   ACCESS MODES
RECLAIM POLICY   STATUS   CLAIM              STORAGECLASS   REASON   AGE
pvc-8a814d62-bd58-4253-b0d1-82f2885db671   1Gi        RWO
Delete           Bound    default/san-pvc    ontap-san               10s
```

Passo 3: Defina um pod que prende o PVC

Fixe o PV a um pod para que ele seja redimensionado. Há dois cenários ao redimensionar um FC PV:

- Se o PV estiver conetado a um pod, o Trident expande o volume no back-end de armazenamento, refaz o dispositivo e redimensiona o sistema de arquivos.
- Ao tentar redimensionar um PV não anexado, o Trident expande o volume no back-end de armazenamento. Depois que o PVC é ligado a um pod, o Trident refaz o dispositivo e redimensiona o sistema de arquivos. Em seguida, o Kubernetes atualiza o tamanho do PVC após a operação de expansão ter sido concluída com sucesso.

Neste exemplo, é criado um pod que usa o `san-pvc`.

```
 kubectl get pod
NAME         READY   STATUS    RESTARTS   AGE
ubuntu-pod   1/1     Running   0          65s

 kubectl describe pvc san-pvc
Name:          san-pvc
Namespace:     default
StorageClass:  ontap-san
Status:        Bound
Volume:        pvc-8a814d62-bd58-4253-b0d1-82f2885db671
Labels:        <none>
Annotations:   pv.kubernetes.io/bind-completed: yes
               pv.kubernetes.io/bound-by-controller: yes
               volume.beta.kubernetes.io/storage-provisioner:
csi.trident.netapp.io
Finalizers:    [kubernetes.io/pvc-protection]
Capacity:      1Gi
Access Modes:  RWO
VolumeMode:    Filesystem
Mounted By:    ubuntu-pod
```

Passo 4: Expanda o PV

Para redimensionar o PV que foi criado de 1Gi a 2Gi, edite a definição de PVC e atualize o `spec.resources.requests.storage` para 2Gi.

```
kubectl edit pvc san-pvc
```

```
# Please edit the object below. Lines beginning with a '#' will be
ignored,
# and an empty file will abort the edit. If an error occurs while saving
this file will be
# reopened with the relevant failures.
#
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    pv.kubernetes.io/bind-completed: "yes"
    pv.kubernetes.io/bound-by-controller: "yes"
    volume.beta.kubernetes.io/storage-provisioner: csi.trident.netapp.io
  creationTimestamp: "2019-10-10T17:32:29Z"
  finalizers:
  - kubernetes.io/pvc-protection
  name: san-pvc
  namespace: default
  resourceVersion: "16609"
  selfLink: /api/v1/namespaces/default/persistentvolumeclaims/san-pvc
  uid: 8a814d62-bd58-4253-b0d1-82f2885db671
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 2Gi
 # ...
```

Etapa 5: Validar a expansão

Você pode validar a expansão trabalhada corretamente verificando o tamanho do PVC, PV e o volume Trident:

```
kubectl get pvc san-pvc
NAME      STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS   AGE
san-pvc   Bound    pvc-8a814d62-bd58-4253-b0d1-82f2885db671   2Gi
RWO            ontap-san      11m
```

```
kubectl get pv
NAME                                       CAPACITY   ACCESS MODES
RECLAIM POLICY   STATUS   CLAIM              STORAGECLASS   REASON   AGE
pvc-8a814d62-bd58-4253-b0d1-82f2885db671   2Gi        RWO
Delete           Bound    default/san-pvc    ontap-san               12m
tridentctl get volumes -n trident
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
|                   NAME                  |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
| pvc-8a814d62-bd58-4253-b0d1-82f2885db671 | 2.0 GiB | ontap-san     |
block    | a9b7bfff-0505-4e31-b6c5-59f492e02d33 | online | true    |
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
```

Expandir um volume NFS

O Trident suporta a expansão de volume para PVs NFS provisionados em `ontap-nas` , `ontap-nas-economy` , `ontap-nas-flexgroup` , e `azure-netapp-files` back-ends.

Etapa 1: Configure o StorageClass para dar suporte à expansão de volume

Para redimensionar um PV NFS, o administrador primeiro precisa configurar a classe de armazenamento para permitir a expansão de volume definindo o `allowVolumeExpansion` campo para `true`:

```
cat storageclass-ontapnas.yaml
```

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontapnas
provisioner: csi.trident.netapp.io
parameters:
  backendType: ontap-nas
allowVolumeExpansion: true
```

Se você já criou uma classe de armazenamento sem essa opção, você pode simplesmente editar a classe de armazenamento existente usando `kubectl edit storageclass` para permitir a expansão de volume.

Etapa 2: Crie um PVC com o StorageClass que você criou

```
cat pvc-ontapnas.yaml
```

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: ontapnas20mb
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 20Mi
  storageClassName: ontapnas
```

A Trident deve criar um PV NFS de 20 MiB para este PVC:

```
kubectl get pvc
NAME           STATUS    VOLUME
CAPACITY     ACCESS MODES   STORAGECLASS    AGE
ontapnas20mb   Bound    pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7   20Mi
RWO            ontapnas        9s

kubectl get pv pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7
NAME                                       CAPACITY   ACCESS MODES
RECLAIM POLICY   STATUS   CLAIM                   STORAGECLASS    REASON
AGE
pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7   20Mi       RWO
Delete           Bound    default/ontapnas20mb   ontapnas
2m42s
```

Passo 3: Expanda o PV

Para redimensionar o PV de 20 MiB recém-criado para 1 GiB, edite o PVC e defina `spec.resources.requests.storage` para 1 GiB:

```
kubectl edit pvc ontapnas20mb
```

```
# Please edit the object below. Lines beginning with a '#' will be
ignored,
# and an empty file will abort the edit. If an error occurs while saving
this file will be
# reopened with the relevant failures.
#
apiVersion: v1
kind: PersistentVolumeClaim
```

```
metadata:
  annotations:
    pv.kubernetes.io/bind-completed: "yes"
    pv.kubernetes.io/bound-by-controller: "yes"
    volume.beta.kubernetes.io/storage-provisioner: csi.trident.netapp.io
  creationTimestamp: 2018-08-21T18:26:44Z
  finalizers:
  - kubernetes.io/pvc-protection
  name: ontapnas20mb
  namespace: default
  resourceVersion: "1958015"
  selfLink: /api/v1/namespaces/default/persistentvolumeclaims/ontapnas20mb
  uid: c1bd7fa5-a56f-11e8-b8d7-fa163e59eaab
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
# ...
```

Etapa 4: Validar a expansão

Você pode validar o redimensionamento trabalhado corretamente verificando o tamanho do PVC, PV e o volume Trident:

```
kubectl get pvc ontapnas20mb
NAME           STATUS   VOLUME
CAPACITY   ACCESS MODES   STORAGECLASS    AGE
ontapnas20mb   Bound    pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7   1Gi
RWO            ontapnas        4m44s

kubectl get pv pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7
NAME                                       CAPACITY   ACCESS MODES
RECLAIM POLICY   STATUS   CLAIM                  STORAGECLASS    REASON
AGE
pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7   1Gi        RWO
Delete           Bound    default/ontapnas20mb   ontapnas
5m35s

tridentctl get volume pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7 -n trident
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
|                  NAME                   |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+-----------------------------------------+---------+---------------
```

```
+----------+----------------------------------------+--------+---------+
| pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7 | 1.0 GiB | ontapnas      |
file     | c5a6f6a4-b052-423b-80d4-8fb491a14a22 | online | true    |
+------------------------------------------+---------+--------------
+----------+----------------------------------------+--------+---------+
```

Entenda os limites do subsistema RWX NVMe

ReadWriteMany (RWX) volumes que utilizam o protocolo NVMe têm um limite de escalabilidade de 64 nós por volume. A seguir, são apresentadas as limitações, explicada a arquitetura do subsistema NVMe envolvido e descritas as etapas necessárias para a resolução do problema.

Entenda o limite de 64 nós

Se você planeja usar ReadWriteMany (RWX) volumes com o protocolo NVMe, um único volume RWX NVMe não pode ser montado por mais de 64 nós em um cluster Kubernetes.

Não agende cargas de trabalho que montem o mesmo RWX NVMe PersistentVolumeClaim em mais de 64 nós.

Essa limitação se aplica somente a volumes RWX que utilizam o protocolo NVMe.

Entenda os modelos de subsistema NVMe

Modelo de subsistema por volume (Trident releases anteriores a 26.02)

Nas versões do Trident anteriores à 26.02, os volumes RWX NVMe são provisionados usando um modelo de subsistema por volume. Cada volume RWX NVMe é mapeado para seu próprio subsistema NVMe dedicado no ONTAP.

Este modelo é simples, mas possui um limite de escalabilidade inferior. Em clusters Kubernetes de grande porte, os limites dos controladores de subsistema são atingidos rapidamente porque cada volume RWX consome um subsistema dedicado.

Modelo de super-subsistema (introduzido no Trident 26.02)

A partir do Trident 26.02, os volumes RWX NVMe utilizam um modelo de super-subsistema compartilhado. Vários volumes RWX NVMe compartilham o mesmo subsistema NVMe.

Cada super-subsistema suporta até 1024 namespaces (volumes). Esse modelo melhora significativamente a escalabilidade para cargas de trabalho RWX e reduz a probabilidade de atingir os limites do subsistema ONTAP.

Cada volume RWX NVMe suporta até 64 nós.

Identifique sintomas de erro

Se você criar ou anexar volumes RWX NVMe em grande escala, poderá observar erros semelhantes aos seguintes:

```
Maximum number of controllers reached. No more controllers can be created.
```

Este erro indica que o limite do controlador do subsistema NVMe do ONTAP foi atingido.

Resolver erros de limite do subsistema

Para superar as limitações de subsistemas por volume e aproveitar as vantagens do modelo de super-subsistema, atualize para Trident 26.02 ou posterior.

Atualize Trident para aplicar o modelo de super-subsistema

Para aplicar o modelo de super-subsistema para volumes RWX NVMe:

1. Atualize Trident para a versão 26.02 ou posterior.
2. Reduza para zero réplicas todos os pods que usam volumes RWX NVMe.
3. Verifique se nenhuma carga de trabalho está usando ativamente volumes RWX NVMe.
4. Aumente a escala dos pods novamente.

Essa sequência de reinicialização garante que os volumes RWX NVMe sejam conectados usando o modelo de super-subsistema.

- Essa limitação se aplica somente a volumes RWX que utilizam o protocolo NVMe.
- O limite de 64 nós se aplica por volume RWX NVMe.
- Outros modos de acesso e outros protocolos não são afetados.

Escalabilidade do controlador

Trident aprimora a escalabilidade do controlador por meio do aumento da simultaneidade em vários drivers de armazenamento. Ao habilitar a escalabilidade do controlador, o Trident processa as operações de armazenamento em paralelo, em vez de serializá-las, o que aumenta a taxa de transferência em ambientes Kubernetes com muitas operações simultâneas.

Antes de implementar, determine quais drivers do Trident oferecem suporte à escalabilidade do controlador na disponibilidade geral e quais drivers estão disponíveis como prévia técnica no Trident 26.10. Isso ajuda você a tomar decisões de implementação mais informadas e a gerenciar riscos. A escalabilidade do controlador está desativada por padrão.

Conceitos e definições principais

Escalabilidade do controlador

A escalabilidade do controlador refere-se à capacidade do controlador Trident de processar múltiplas operações de storage em paralelo, em vez de serializá-las por trás de um único bloqueio. Essas operações incluem criação, exclusão e redimensionamento de volumes; criação e exclusão de snapshots; publicação e cancelamento de publicação de volumes; e gerenciamento de backend.

Ao habilitar a escalabilidade do controlador, as operações em diferentes volumes e backends são executadas simultaneamente. Isso aumenta a taxa de transferência e reduz o tempo de operação completamente em

ambientes com um grande número de operações simultâneas de PersistentVolumeClaim e VolumeSnapshot.

Comportamento padrão (modo serial)

Por padrão, o controlador Trident processa as operações uma de cada vez. Cada solicitação de criação, exclusão, redimensionamento ou snapshot é concluída antes que o Trident inicie a próxima. O modo serial é o padrão compatível para todas as instalações e não requer configuração.

O modo serial é suficiente para a maioria das cargas de trabalho. Habilite a escalabilidade do controlador somente quando o volume de operações criar um acúmulo no processamento serial.

Suporte à escalabilidade do controlador

Trident oferece suporte à escalabilidade em diferentes níveis de maturidade para diferentes drivers de armazenamento.

Disponibilidade geral

Os seguintes drivers oferecem suporte à escalabilidade em disponibilidade geral no Trident 26.10:

- `ontap-san`
- `ontap-nas`
- `ontap-nas-economy`
- `ontap-nas-flexgroup`
- `ontap-san-economy`
- `google-cloud-netapp-volumes`
- `azure-netapp-files`
- `solidfire-san`

OBSERVAÇÃO	Os `google-cloud-netapp-volumes` e `google-cloud-netapp-volumes-san` drivers são diferentes. Trident suporta apenas `google-cloud-netapp-volumes`. Não use `google-cloud-netapp-volumes-san` em configurações ou exemplos de backend.

Prévia técnica

O seguinte driver oferece suporte à escalabilidade como um recurso de pré-visualização técnica no Trident 26.10:

- `asa-r2` (SAN e NVMe)

Este driver possui as seguintes limitações:

- A concorrência do controlador está disponível apenas para avaliação e teste.
- O comportamento pode mudar de uma versão para outra.
- NetApp não recomenda o uso em ambientes de produção.

Como enableConcurrency funciona

Quando você define `enableConcurrency` como `true`, o Trident aplica processamento simultâneo em todos os backends que o controlador gerencia. A configuração se aplica a todos os backends de uma só vez. Você não pode habilitá-la para backends ou drivers individuais.

Cada backend configurado deve usar um driver presente na lista de disponibilidade geral ou na lista de pré-visualização técnica. Se algum backend usar um driver não suportado, Trident não inicia e Trident não adiciona um backend que use um driver não suportado.

Para restaurar a inicialização, remova ou reconfigure qualquer backend que utilize um driver não suportado ou desative a escalabilidade do controlador.

Antes de ativar

IMPORTANTE	Antes de habilitar a escalabilidade do controlador, confirme que cada backend configurado usa um driver presente na lista de disponibilidade geral ou na lista de prévia técnica. Se algum backend usar um driver não compatível, o Trident não será iniciado após você habilitar o recurso.

Utilize a tabela a seguir para decidir se deseja habilitar a escalabilidade do controlador.

Se o seu ambiente	Então
Utiliza apenas drivers compatíveis e apresenta um acúmulo de operações do controlador	Habilite a escalabilidade do controlador.
Utiliza qualquer driver não suportado	Mantenha o modo serial padrão. Não habilite a escalabilidade do controlador.
Lida com baixo volume de operações sem acúmulo de tarefas	Mantenha o modo serial padrão.

Habilitar escalabilidade do controlador

A opção de configuração `enableConcurrency` controla a escalabilidade do controller. Você deve habilitar explicitamente essa opção durante a instalação do Trident ou ao atualizar uma implantação existente.

Desdobramento do operador Trident

Para habilitar a escalabilidade do controlador com o operador Trident, defina `enableConcurrency` como `true` no `TridentOrchestrator` recurso personalizado (CR).

Nova instalação

Crie ou edite o `TridentOrchestrator`CR e defina `enableConcurrency` como `true`:

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  namespace: trident
```

```
  enableConcurrency: true
```

Aplicar o CR:

```
kubectl apply -f tridentorchestrator_cr.yaml
```

Instalação existente

Corrija a `TridentOrchestrator`CR existente para permitir a escalabilidade do controlador:

```
kubectl patch torc trident --type=merge -p
'{"spec":{"enableConcurrency":true}}'
```

Verifique se Trident aplicou a configuração:

```
kubectl get torc trident -o
jsonpath='{.status.currentInstallationParams.enableConcurrency}'
```

Implantação Helm

Para habilitar a escalabilidade do controlador com o Helm, defina o valor `enableConcurrency` como `true`.

Nova instalação

```
helm install trident netapp-trident/trident-operator --namespace trident
--create-namespace --set enableConcurrency=true
```

Instalação existente

```
helm upgrade trident netapp-trident/trident-operator --namespace trident
--set enableConcurrency=true
```

Alternativamente, defina `enableConcurrency` para `true` em um arquivo `values.yaml` personalizado:

```
# values.yaml
enableConcurrency: true
```

Em seguida, instale ou atualize usando o arquivo de valores:

```
helm install trident netapp-trident/trident-operator --namespace trident
```

```
--create-namespace -f values.yaml
```

implantação do tridentctl

Para habilitar a escalabilidade do controlador com `tridentctl`, passe a flag `--enable-concurrency` durante a instalação.

Nova instalação

```
tridentctl install -n trident --enable-concurrency
```

Instalação existente

Para habilitar a escalabilidade do controlador em uma implantação `tridentctl` existente, desinstale Trident e reinstale-o com a flag:

```
tridentctl uninstall -n trident
tridentctl install -n trident --enable-concurrency
```

Verifique se a escalabilidade do controlador está ativada

Após habilitar a escalabilidade do controlador, verifique se o controlador Trident está sendo executado com a concorrência ativada. Verifique os logs do pod do controlador:

```
kubectl logs -n trident deploy/trident-controller | grep -i concurrency
```

A saída inclui uma entrada de log que confirma que a concorrência está habilitada.

Comportamento de concorrência

Quando a escalabilidade do controlador está ativada, o controlador Trident aplica o seguinte comportamento:

- Trident substitui o bloqueio global único por bloqueios granulares, por recurso.
- Trident serializa operações que modificam o mesmo recurso para manter a consistência de dados.
- Operações que apenas leem de um recurso são executadas simultaneamente com outras operações de leitura nesse recurso.
- Trident limita as solicitações simultâneas da API ONTAP a 20 por LIF de gerenciamento para evitar sobrecarga dos sistemas de armazenamento de backend.
- Se vários backends compartilharem a mesma LIF de gerenciamento, eles compartilharão esse limite de 20 solicitações.

Ressalvas e limitações

As seguintes considerações aplicam-se à escalabilidade do controlador no Trident 26.10:

- A escalabilidade do controlador suporta apenas os drivers nas listas de disponibilidade geral e de pré-

visualização técnica. Para obter detalhes, consulte Antes de ativar.

- O controlador Trident gerencia a concorrência internamente. Esta versão não oferece limites de concorrência configuráveis pelo usuário.
- A taxa de transferência geral depende do driver de armazenamento em uso, da capacidade de resposta do backend e do desempenho do servidor da API do Kubernetes.
- A alta concorrência pode aumentar a carga nos sistemas de storage backend.
- O comportamento de escalabilidade do controlador não é idêntico em todos os drivers.
- O driver de pré-visualização técnica pode apresentar desempenho inconsistente sob alta carga e pode mudar de comportamento entre as versões.
- A depuração de operações simultâneas pode ser mais complexa devido à execução paralela. Métricas e registros podem mostrar a saída de operações intercaladas.

Recomendações

Antes de ativar a escalabilidade do controlador, conclua as seguintes etapas:

1. Confirme se cada backend configurado utiliza um driver presente na lista de disponibilidade geral ou de prévia técnica.
2. Teste a alteração em um cluster de não produção antes de aplicá-la em produção.
3. Verifique se a escalabilidade do controlador está ativada após você aplicar a alteração.

Aplique as seguintes recomendações gerais ao trabalhar com escalabilidade de controladora:

- Utilize drivers de disponibilidade geral para ambientes de produção que exigem alta escalabilidade.
- Avalie o driver de pré-visualização técnica somente em ambientes não produtivos.
- Monitore o desempenho do backend e do controlador quando você opera em grande escala.
- Não assuma a ordem das operações em scripts de automação.

Expansão de volume automática

A expansão de volume permite que os Volumes Persistentes provisionados pelo Trident cresçam automaticamente quando a capacidade utilizada atinge um limite definido. Essa funcionalidade reduz a sobrecarga operacional e ajuda a evitar interrupções nos aplicativos causadas pelo esgotamento da capacidade. A expansão de volume automática é implementada usando Autogrow Policies. Uma Autogrow Policy define:

- O limite de utilização que desencadeia a expansão
- A quantidade pela qual o volume cresce
- O tamanho máximo que o volume pode atingir

As políticas Autogrow do Trident expandem o PVC. Isso difere de "Dimensionamento automático de volume do ONTAP", que aumenta o FlexVol de suporte sem alterar o tamanho do PVC.

OBSERVAÇÃO	Os volumes aumentam de tamanho automaticamente quando o limite de utilização definido é excedido. Os volumes nunca são reduzidos automaticamente.

Requisitos

Antes de configurar a expansão automática de volume, certifique-se de que os seguintes requisitos sejam atendidos:

- Trident 26.02 ou posterior
- Permissões de controle de acesso baseadas em funções para criar `TridentAutogrowPolicy` recursos personalizados
- StorageClasses configurado com `allowVolumeExpansion: true`
- Protocolos ONTAP suportados:
  - Network File System (NFS)
  - Internet Small Computer Systems Interface (iSCSI)
  - Memória Não Volátil Express (NVMe)
  - Protocolo Fibre Channel (FCP)

Limitações

- Os volumes de bloco bruto do ONTAP Non-Volatile Memory Express anteriores ao ONTAP 9.16.1 não suportam expansão automática.
- Para volumes de rede de área de armazenamento, se `growthAmount` for menor ou igual a 50 mebibytes, Trident aumenta automaticamente o valor para 51 mebibytes antes de redimensionar, desde que o tamanho resultante não exceda `maxSize`.
- Em ambientes já existentes, a expansão automática pode não funcionar para determinados volumes existentes devido ao comportamento de migração de publicação de volume.
- Quando um volume atinge `maxSize`, nenhuma expansão adicional ocorre.
- Protocolos suportados para expansão automática de volume:
  - Network File System (NFS)
  - Internet Small Computer Systems Interface (iSCSI)
  - Memória Não Volátil Express (NVMe)
  - Protocolo Fibre Channel (FCP)

Provisionar volumes com política de crescimento automático

A política de volume com crescimento automático pode ser configurada em dois níveis:

- Nível da classe de armazenamento: define o padrão para todos os volumes (usando anotação)
- Nível PVC: substitui o padrão da classe de armazenamento (usando anotação)

Criar uma política de crescimento automático

As políticas de crescimento automático permitem a expansão automática do volume quando os volumes atingem um limite de capacidade definido.

Certifique-se de ter:

- Trident 26.02 ou posterior instalado

- Permissões de controle de acesso baseadas em funções para criar `TridentAutogrowPolicy` recursos
- Compreensão dos requisitos de crescimento da carga de trabalho

Uma Autogrow Policy define como os volumes se expandem automaticamente quando atingem um limite de capacidade definido.

Você pode criar políticas de crescimento automático em qualquer ponto do seu fluxo de trabalho:

- Antes de StorageClasses e volumes serem criados
- Após StorageClasses existirem
- Após o provisionamento dos volumes

Essa flexibilidade permite que você introduza a expansão automática sem recriar os recursos existentes.

Especificações da política de volume com crescimento automático

As políticas de crescimento automático são recursos personalizados do Kubernetes definidos da seguinte forma:

Campo	Descrição	Formato	Obrigatório	Exemplo	Padrão
nome	Identificador de política exclusivo	Cadeia de carateres	Sim	política-db-produção	Nenhum
usedThreshold	Percentual de capacidade que desencadeia a expansão	String de porcentagem	Sim	"80%"	Nenhum
growthAmount	Quantidade a crescer quando o limite for atingido	Porcentagem ou tamanho	Não	"10%" ou "5Gi"	"10%"
maxSize	Limite máximo de tamanho do volume	Quantidade de Kubernetes	Não	"500Gi"	Ilimitado

Criar uma política de crescimento automático

Passos

1. Crie um arquivo YAML que defina sua Autogrow Policy:

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: standard-autogrow
spec:
  usedThreshold: "80%"
  growthAmount: "10%"
  maxSize: "500Gi"
```

2. Aplique a política ao seu cluster:

```
kubectl apply -f autogrow-policy.yaml
```

3. Verifique se a política foi criada:

```
kubectl get tridentautogrowpolicy standard-autogrow
```

Resultado esperado

```
NAME                USED THRESHOLD   GROWTH AMOUNT   STATE
standard-autogrow   80%              10%             Success
```

Estados de política

Depois que você cria uma política, Trident valida a especificação e atribui um dos seguintes estados:

Estado	Descrição	Ação necessária
Sucesso	A política é validada e está pronta para uso.	Nenhum.
Falha	Foram detectados erros de validação.	Revise e corrija a especificação.
Excluindo	A exclusão está em andamento.	Aguarde a conclusão.

Associe uma política a um StorageClass

Você pode associar uma Autogrow Policy a um StorageClass usando a `trident.netapp.io/autogrowPolicy` anotação. Todos os volumes provisionados a partir desse StorageClass herdam a política.

OBSERVAÇÃO O StorageClass deve ter `allowVolumeExpansion: true`.

Passos

1. Crie ou modifique uma StorageClass com a anotação de Autogrow Policy:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-gold
  annotations:
    trident.netapp.io/autogrowPolicy: "production-db-policy"
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
  fsType: "ext4"
```

```
allowVolumeExpansion: true
```

2. Aplique o StorageClass:

```
kubectl apply -f storageclass.yaml
```

3. Verifique a anotação:

```
kubectl get storageclass ontap-gold -o
jsonpath='{.metadata.annotations.trident\.netapp\.io/autogrowPolicy}'
```

Resultado esperado

```
production-db-policy
```

Precedência de política

Quando as anotações da Política de Crescimento Automático são definidas tanto em um StorageClass quanto em um PVC, Trident aplica as seguintes regras de precedência:

1. **A anotação PVC tem prioridade.** Se um PVC definir `trident.netapp.io/autogrowPolicy`, esse valor será sempre usado.
2. **StorageClass a anotação só se aplica quando o PVC não possui nenhuma anotação.**
3. **Se nenhum dos dois tiver a anotação,** nenhuma Autogrow Policy será aplicada.

StorageClass anotação	Anotação PVC	Comportamento eficaz
`trident.netapp.io/autogrowPolicy: standard-agp`	Não definido	Usos `standard-agp`.
`trident.netapp.io/autogrowPolicy: standard-agp`	`trident.netapp.io/autogrowPolicy: logs-policy`	Usa `logs-policy` (PVC substitui StorageClass).
`trident.netapp.io/autogrowPolicy: standard-agp`	`trident.netapp.io/autogrowPolicy: "none"`	Sem política de crescimento automático (PVC desativa o autogrow).
Não definido	`trident.netapp.io/autogrowPolicy: dev-policy`	Usos `dev-policy`.
Não definido	Não definido	Sem política de autogrow.

Exemplos de configuração

Os exemplos a seguir mostram configurações comuns de Autogrow Policy para diferentes casos de uso.

Política conservadora para bancos de dados de produção

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: production-db-policy
spec:
  usedThreshold: "75%"
  growthAmount: "20%"
  maxSize: "5Ti"
```

Armazenamento de logs com incrementos de crescimento fixos

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: log-storage-policy
spec:
  usedThreshold: "90%"
  growthAmount: "10Gi"
  maxSize: "100Gi"
```

Política mínima com valores padrão

Quando você omite `growthAmount` e `maxSize`, Trident usa os padrões (`10%` de crescimento, tamanho ilimitado):

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: logs-policy
spec:
  usedThreshold: "85%"
```

Política com um maxSize personalizado e growthAmount padrão

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: default-ga-policy
spec:
  usedThreshold: "70%"
  maxSize: "100Gi"
```

Crescimento agressivo com maxSize ilimitado

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: aggressive-growth-policy
spec:
  usedThreshold: "80%"
  growthAmount: "150%"
```

Política com percentuais fracionários

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: precise-policy
spec:
  usedThreshold: "80.28%"
  growthAmount: "10.65%"
  maxSize: "100Gi"
```

OBSERVAÇÃO	São aceitas porcentagens fracionárias. Se você especificar mais de três casas decimais, Trident arredonda o valor para três casas decimais.

NAS StorageClass com Autogrow

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-nas-autogrow
  annotations:
    trident.netapp.io/autogrowPolicy: "standard-autogrow"
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-nas"
  fsType: "ext4"
allowVolumeExpansion: true
```

SAN StorageClass com Autogrow

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
```

```
  name: database-storage
  annotations:
    trident.netapp.io/autogrowPolicy: "production-db-policy"
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
  fsType: "ext4"
allowVolumeExpansion: true
```

Configurar o dimensionamento automático de volume do ONTAP

Configure o Trident para habilitar o dimensionamento automático de volume do ONTAP nos volumes FlexVol que dão suporte a volumes persistentes. Quando o espaço usado atinge um limite configurado, o ONTAP aumenta automaticamente o FlexVol até um tamanho máximo. Isso fornece capacidade adicional para snapshots e sobrecarga de storage sem redimensionar manualmente o FlexVol.

O dimensionamento automático de volume do ONTAP aumenta apenas o FlexVol de suporte; ele não altera o tamanho do PersistentVolumeClaim (PVC). Isso difere de "Políticas de crescimento automático do Trident", que expandem o PVC.

O Trident aplica as configurações de dimensionamento automático de backend quando cria, clona ou redimensiona um volume e quando importa um volume que gerencia.

Requisitos

- Trident 26.10 ou posterior.
- Um dos seguintes drivers:
  - `ontap-san` com iSCSI, NVMe ou Fibre Channel Protocol (FCP).
  - `ontap-san-economy` com iSCSI.
- ONTAP no local:
  - Para backends que utilizam as APIs REST do ONTAP, ONTAP 9.12.1 ou posterior. Os backends NVMe devem utilizar as APIs REST do ONTAP.
  - Para backends que utilizam ONTAPI (ZAPI), qualquer versão do ONTAP suportada pelo Trident. Consulte "Requisitos da Trident".

Limitações

- O dimensionamento automático de volume do ONTAP não reduz FlexVols.
- O dimensionamento automático é configurado por backend. Crie backends separados para usar configurações diferentes para cargas de trabalho diferentes.
- O dimensionamento automático de volume do ONTAP não é compatível com os drivers NAS do ONTAP, os sistemas ASA r2, o NetApp AFX ou outros sistemas ONTAP desagregados, nem com o Google Cloud NetApp Volumes. A criação do backend falha nessas configurações.
- A replicação do TridentMirrorRelationship e as configurações do MetroCluster não são compatíveis com o dimensionamento automático de volume do ONTAP. O Trident não impede essas configurações.

- Para `ontap-san-economy`, `maximumSize` deve ser uma porcentagem.
- Para `ontap-san`, um valor absoluto `maximumSize` atua como um limite de tamanho. A criação ou expansão de um PVC falha se o FlexVol subjacente precisar exceder esse limite.
- Um `ontap-san-economy` FlexVol pode conter LUNs para vários PVCs. Cada operação de volume aplica as configurações de dimensionamento automático do backend atual a todo o FlexVol. Se vários backends compartilharem FlexVols, a operação mais recente determinará as configurações deles. Para `ontap-san-economy`, configure apenas `mode: grow` e omita `growThresholdPercent` e `maximumSize` para que todos os backends usem os mesmos padrões.

Opções de dimensionamento automático de volume do ONTAP

Defina essas opções no bloco `ontapAutosize` de nível superior da configuração do backend.

Opção	Descrição	Formato	Obrigatório	Padrão quando `ontapAutosize` está presente	Exemplo
`mode`	Ativa ou desativa o dimensionamento automático. `grow` ativa o dimensionamento automático; `off` desativa-o.	Cadeia de carateres	Não	`off`	`grow` ou `off`
`growThresholdPercent`	Porcentagem de espaço utilizado a partir da qual o ONTAP aumenta o FlexVol. Os valores válidos são de 2 a 99.	Número inteiro	Não	`85`	`90`
`maximumSize`	Tamanho máximo do FlexVol, expresso como uma porcentagem do seu tamanho ou como um tamanho absoluto. Tamanhos absolutos são suportados apenas para `ontap-san`.	Porcentagem ou quantidade do Kubernetes	Não	`"20%"`	`"20%"` ou `"2Ti"`

As configurações padrão se aplicam quando o `ontapAutosize` bloco está presente. Se você omitir o bloco, o Trident não alterará as configurações de dimensionamento automático nos FlexVols.

O Trident converte uma porcentagem `maximumSize` em um tamanho absoluto e a recalcula sempre que o volume é redimensionado. Por exemplo, com `maximumSize: "20%"`, um FlexVol de 100 GiB pode crescer até 120 GiB.

Configurar o dimensionamento automático de volume do ONTAP

1. Adicione o `ontapAutosize` bloco à sua configuração de backend:

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-san
  namespace: trident
```

```
spec:
  version: 1
  storageDriverName: ontap-san
  backendName: ontap-san-autosize
  managementLIF: 10.0.0.1
  svm: svm_san
  sanType: iscsi
  ontapAutosize:
    mode: grow
    growThresholdPercent: 85
    maximumSize: "20%"
  credentials:
    name: backend-tbc-ontap-san-secret
```

2. Crie o backend:

```
kubectl -n trident create -f backend-tbc-ontap-san.yaml
```

3. Após provisionar um volume a partir deste backend, verifique as configurações de dimensionamento automático aplicadas ao volume do Trident:

```
kubectl get tridentvolume <pv-name> -n trident \
  -o jsonpath='{.config.ontapAutosizeMode}
{.config.ontapAutosizeGrowThresholdPercent}
{.config.ontapAutosizeMaximumSize}'
```

Para a configuração acima, o resultado esperado é:

```
grow 85 20%
```

Atualizar as configurações de dimensionamento automático em um backend

Você pode adicionar, alterar ou remover o `ontapAutosize` bloco atualizando o backend. A alteração não atualiza imediatamente os FlexVols existentes:

- Para `ontap-san`, novos volumes, clones e importações gerenciadas usam as novas configurações. Os volumes existentes mantêm suas configurações atuais, inclusive quando são redimensionados.
- Para `ontap-san-economy`, o FlexVol compartilhado recebe as novas configurações na próxima vez em que o Trident cria, redimensiona, clona ou importa um volume nele.

Exemplos de configuração

`ontap-san` com FCP e um tamanho máximo absoluto

```
{
  "version": 1,
  "storageDriverName": "ontap-san",
  "managementLIF": "10.0.0.1",
  "svm": "svm_san",
  "username": "vsadmin",
  "password": "<password>",
  "sanType": "fcp",
  "ontapAutosize": {
    "mode": "grow",
    "growThresholdPercent": 90,
    "maximumSize": "2Ti"
  }
}
```

`ontap-san-economy`

Este exemplo define apenas `mode`, portanto o Trident usa os valores padrão para `growThresholdPercent` e `maximumSize`:

```
{
  "version": 1,
  "storageDriverName": "ontap-san-economy",
  "managementLIF": "10.0.0.1",
  "svm": "svm_san",
  "username": "vsadmin",
  "password": "<password>",
  "ontapAutosize": {
    "mode": "grow"
  }
}
```

Perguntas frequentes

Posso usar o dimensionamento automático de volume do ONTAP com as Trident Autogrow Policies?

Sim. Uma política de crescimento automático do Trident expande o PVC, e o Trident recalcula o máximo do dimensionamento automático do ONTAP durante cada redimensionamento. Se você usar um valor absoluto `maximumSize` com `ontap-san`, defina o `maxSize` da política baixo o suficiente para manter o FlexVol subjacente dentro desse limite. O FlexVol é maior que o PVC porque o Trident adiciona espaço para reserva de snapshot e metadados.

Qual é a diferença entre omitir `ontapAutosize` e definir `mode: off`?

Se você omitir `ontapAutosize`, o Trident não altera as configurações de dimensionamento automático nos FlexVols do ONTAP. Se você definir `mode: off`, o Trident desativa o dimensionamento automático nos FlexVols que ele cria ou atualiza, incluindo clones que herdaram as configurações de dimensionamento automático da origem.

Por que minha expansão de PVC falhou com um `ontapAutosize maximum … is below required FlexVol size` erro?

O volume usa um valor absoluto `maximumSize`, e o PVC expandido requer um FlexVol maior. Um `ontap-san` volume mantém o tamanho máximo com o qual foi criado, portanto atualizar o backend não altera a configuração nos volumes existentes. Para novos volumes, aumente `maximumSize` ou use uma porcentagem.

Por que a criação do volume falhou com um erro de dimensionamento automático?

O ONTAP rejeitou as configurações de dimensionamento automático, e os eventos de PVC incluem o motivo fornecido pelo ONTAP. Por exemplo, o tamanho máximo pode exceder o maior tamanho de FlexVol compatível com o sistema ONTAP. Reduza `maximumSize` no backend; o Trident usa o valor atualizado na próxima vez que tenta criar um volume.

Informações relacionadas

- "Expansão de volume automática usando o Trident Autogrow Policies"
- "Referência de comandos do ONTAP: dimensionamento automático de volume"

Gerenciar políticas de crescimento automático

Após criar políticas de crescimento automático, você pode visualizá-las, atualizá-las e excluí-las. Você também pode monitorar quais volumes usam uma determinada política.

Ver políticas de Autogrow

Listar todas as políticas

Use `kubectl` para listar todas as Autogrow Policies no seu cluster:

```
kubectl get tridentautogrowpolicy
```

Alternativamente, use `tridentctl`:

```
tridentctl get autogrowpolicy
```

Ver detalhes da política

Para visualizar a especificação completa e o status de uma policy:

```
kubectl describe tridentautogrowpolicy production-db-policy
```

Para visualizar uma apólice com seus respectivos volumes em formato YAML:

```
tridentctl get autogrowpolicy production-db-policy -o yaml
```

Atualizar uma política de crescimento automático

Você pode modificar uma política existente para alterar seu limite, quantidade de crescimento ou tamanho máximo. As alterações entram em vigor imediatamente para todos os volumes que utilizam a política.

IMPORTANTE	As alterações afetam todos os volumes que utilizam a política. Sempre que possível, teste as alterações primeiro em um ambiente que não seja de produção.

Passos

1. Edite a política:

   ```
   kubectl edit tridentautogrowpolicy production-db-policy
   ```

2. Modifique os `spec` fields conforme necessário:

   ```
   spec:
     usedThreshold: "75%"     # Changed from 80%
     growthAmount: "20%"      # Changed from 10%
     maxSize: "1Ti"           # Changed from 500Gi
   ```

3. Salve e saia. As alterações entram em vigor imediatamente.

Considerações sobre atualização

- **Efeito imediato:** Todos os volumes que utilizam a política adotam os novos parâmetros na próxima avaliação de crescimento.
- **Não é necessário reiniciar o volume:** As alterações serão aplicadas na próxima operação de crescimento.
- **Teste primeiro:** Valide as alterações em um ambiente que não seja de produção quando possível.
- **Comunique alterações:** Notifique as equipes quando você modificar as políticas compartilhadas.

Excluir uma política de crescimento automático

As políticas de crescimento automático usam proteção de finalizador para evitar exclusão acidental enquanto os volumes as utilizam ativamente.

Passos

1. Excluir a política:

```
kubectl delete tridentautogrowpolicy production-db-policy
```

2. Se os volumes ainda utilizarem a política, a exclusão entra em um estado `Deleting`. Verifique quais volumes são afetados:

```
tridentctl get autogrowpolicy production-db-policy -o yaml
```

3. Remova a política de cada volume afetado. Escolha uma das seguintes opções:

   - **Opção A: desative explicitamente o crescimento automático** definindo a anotação como `"none"`:

```
kubectl annotate pvc <pvc-name> \
  trident.netapp.io/autogrowPolicy="none" \
  --overwrite
```

   - **Opção B: remover a anotação completamente**

```
kubectl annotate pvc <pvc-name> \
  trident.netapp.io/autogrowPolicy-
```

Comportamento de exclusão

Cenário	Comportamento
Nenhum volume utiliza a política	A política é excluída imediatamente.
Os volumes utilizam a política	A política entra em `Deleting` estado. Um finalizador bloqueia a conclusão até que todos os volumes sejam removidos.
Todos os volumes são removidos da política	Os finalizadores são removidos e a policy é excluída.

Monitorar o uso da política Autogrow

Verificar volumes usando uma política

```
tridentctl get autogrowpolicy production-db-policy -o json | jq '.volumes'
```

Descubra qual política um volume utiliza

```
kubectl get pvc database-pvc -o
jsonpath='{.metadata.annotations.trident\.netapp\.io/autogrowPolicy}'
```

Monitorar eventos de políticas

```
kubectl get events --field-selector
involvedObject.kind=TridentAutogrowPolicy
```

Protocolos suportados

Autogrow suporta os seguintes protocolos de armazenamento:

- NFS
- ISCSI
- FCP
- NVMe

OBSERVAÇÃO	Para volumes SAN, se o `growthAmount` configurado for de 50 MiB ou menos, Trident aumenta automaticamente o valor de crescimento para 51 MB na operação de redimensionamento, desde que o tamanho resultante não exceda `maxSize`.

Backends suportados

Autogrow é compatível com backends ONTAP. Autogrow também é compatível com Google Cloud NetApp Volumes (GCNV) para volumes NAS e SAN.

Limitações conhecidas

- **ONTAP Volumes de bloco bruto NVMe:** Volumes criados com versões do ONTAP anteriores à 9.16.1 não suportam autogrow.
- **Volumes existentes (implantações brownfield):** Autogrow pode não funcionar para volumes existentes, mesmo que uma política de Autogrow válida seja aplicada. Essa limitação ocorre devido a uma migração em andamento das publicações de volumes. Para confirmar que a migração foi concluída, verifique os logs do controlador Trident em busca de `"Migration completed"` mensagens.

Perguntas frequentes

Quando o Trident avalia o limiar?

Trident monitora continuamente o uso do volume. Quando a capacidade utilizada ultrapassa o `usedThreshold`, Trident cria uma solicitação interna de redimensionamento e expande o volume pela configuração de `growthAmount`.

Por exemplo, esta política aciona a expansão em 80% da capacidade e aumenta o volume em 10% cada vez, até um máximo de 500 GiB:

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: standard-autogrow
spec:
```

```
  usedThreshold: "80%"
  growthAmount: "10%"
  maxSize: "500Gi"
```

Posso aplicar uma política depois que os volumes já foram provisionados?

Sim. Você pode criar uma Política de Crescimento Automático a qualquer momento e aplicá-la a PVCs existentes adicionando ou atualizando a `trident.netapp.io/autogrowPolicy` anotação. Você não precisa recriar o PVC ou o StorageClass.

Para aplicar uma política a um PVC existente:

```
kubectl annotate pvc <pvc-name> \
  trident.netapp.io/autogrowPolicy="production-db-policy" \
  --overwrite
```

Para aplicar uma política a um StorageClass existente:

```
kubectl annotate storageclass ontap-gold \
  trident.netapp.io/autogrowPolicy="production-db-policy" \
  --overwrite
```

O que acontece se eu definir uma Autogrow Policy tanto no StorageClass quanto no PVC?

A anotação do PVC sempre tem precedência. Se um PVC tiver a `trident.netapp.io/autogrowPolicy` anotação, Trident usa esse valor independentemente do que o StorageClass especifica. Consulte "Precedência de política" para obter detalhes.

Por exemplo, dado este StorageClass:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-gold
  annotations:
    trident.netapp.io/autogrowPolicy: "standard-agp"
provisioner: csi.trident.netapp.io
allowVolumeExpansion: true
```

E este PVC que se sobrepõe à política de StorageClass:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
```

```
  name: database-pvc
  annotations:
    trident.netapp.io/autogrowPolicy: "logs-policy"
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 50Gi
  storageClassName: ontap-gold
```

Trident usa `logs-policy` para `database-pvc`, não `standard-agp`.

Como faço para desativar o autogrow para um volume específico?

Defina a anotação PVC como `"none"`. Essa configuração substitui qualquer política de nível de StorageClass para esse volume:

```
kubectl annotate pvc <pvc-name> \
  trident.netapp.io/autogrowPolicy="none" \
  --overwrite
```

Você pode verificar se o autogrow está desativado:

```
kubectl get pvc <pvc-name> -o jsonpath
='{.metadata.annotations.trident\.netapp\.io/autogrowPolicy}'
```

Resultado esperado

```
none
```

O que acontece quando um volume atinge maxSize?

Trident interrompe a expansão do volume. Ele não cria mais solicitações de redimensionamento para esse volume, mesmo que o uso continue a aumentar além do `usedThreshold`.

Por exemplo, com essa política, Trident para de aumentar o volume depois que ele atinge 100 GiB:

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: capped-policy
spec:
  usedThreshold: "90%"
  growthAmount: "10Gi"
```

```
  maxSize: "100Gi"
```

Para permitir crescimento ilimitado, omita `maxSize` ou defina como `0`:

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: unlimited-policy
spec:
  usedThreshold: "85%"
  growthAmount: "10%"
```

Posso alterar uma política sem reiniciar os volumes?

Sim. Quando você atualiza uma política, todos os volumes que usam essa política adotam os novos parâmetros na próxima avaliação de crescimento. Não é necessário reiniciar os volumes.

Para atualizar uma política existente:

```
kubectl edit tridentautogrowpolicy production-db-policy
```

Modifique os campos conforme necessário:

```
spec:
  usedThreshold: "75%"     # Changed from 80%
  growthAmount: "20%"      # Changed from 10%
  maxSize: "1Ti"           # Changed from 500Gi
```

Salve e saia. Verifique a política atualizada:

```
kubectl get tridentautogrowpolicy production-db-policy
```

Resultado esperado

```
NAME                    USED THRESHOLD   GROWTH AMOUNT   STATE
production-db-policy    75%              20%             Success
```

Por que minha policy está em estado de falha?

Um `Failed` estado indica que a especificação da política contém erros de validação. Execute o seguinte comando para visualizar os detalhes do erro:

```
kubectl describe tridentautogrowpolicy <policy-name>
```

As causas comuns incluem um `usedThreshold` inválido (deve estar entre 1–99%), um `growthAmount` que excede `maxSize`, ou um formato de quantidade do Kubernetes inválido. Corrija a especificação e reaplique:

```
kubectl apply -f autogrow-policy.yaml
```

Por que não consigo excluir uma política?

As políticas utilizam proteção de finalização. Se os volumes ainda utilizarem a política, a exclusão entra em um estado `Deleting` e aguarda até que você remova todos os volumes da política.

Identifique os volumes afetados:

```
tridentctl get autogrowpolicy production-db-policy -o yaml
```

Em seguida, remova a anotação de cada PVC:

```
# Option A: Explicitly disable autogrow
kubectl annotate pvc <pvc-name> \
  trident.netapp.io/autogrowPolicy="none" \
  --overwrite

# Option B: Remove the annotation entirely
kubectl annotate pvc <pvc-name> \
  trident.netapp.io/autogrowPolicy-
```

Após você remover todos os volumes, Trident libera o finalizador e exclui a política.

Quais backends o autogrow suporta?

O Autogrow é compatível com backends ONTAP. Ele também é compatível com Google Cloud NetApp Volumes (GCNV) para volumes NAS e SAN.

O Autogrow é compatível com os protocolos NFS, iSCSI, FCP e NVMe. Volumes de bloco bruto NVMe requerem ONTAP 9.16.1 ou posterior.

Os volumes existentes em implantações brownfield podem exigir que a migração da publicação de volume seja concluída antes que o autogrow entre em vigor. Para verificar o status da migração, verifique os logs do controlador Trident:

```
kubectl logs -l app=trident-controller -n trident | grep "Migration
completed"
```

Os seguintes exemplos de StorageClass mostram o autogrow configurado para backends NAS e SAN:

Backend NAS

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-nas-autogrow
  annotations:
    trident.netapp.io/autogrowPolicy: "standard-autogrow"
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-nas"
  fsType: "ext4"
allowVolumeExpansion: true
```

Backend SAN

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: database-storage
  annotations:
    trident.netapp.io/autogrowPolicy: "production-db-policy"
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
  fsType: "ext4"
allowVolumeExpansion: true
```

Qual é o valor mínimo de crescimento para volumes SAN?

Para volumes SAN, o crescimento mínimo efetivo é de 51 MB. Se você configurar um `growthAmount` de 50 MiB ou menos, Trident aumenta automaticamente o crescimento para 51 MB na operação de redimensionamento.

Por exemplo, esta política define um `growthAmount` de `"40Mi"`, mas Trident aplica um crescimento de 51 MB para qualquer volume SAN que a utilize:

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: san-minimal-policy
spec:
  usedThreshold: "85%"
  growthAmount: "40Mi"
```

```
  maxSize: "100Gi"
```

Para evitar esse ajuste automático, defina `growthAmount` para um valor superior a 50 MiB:

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: san-policy
spec:
  usedThreshold: "85%"
  growthAmount: "100Mi"
  maxSize: "500Gi"
```

Importar volumes do Trident

Você pode importar volumes de armazenamento existentes como um PV do Kubernetes usando `tridentctl import` ou criando uma Persistent Volume Claim (PVC) com anotações de importação do Trident.

Visão geral e considerações

Você pode importar um volume para o Trident para:

- Conteinerize um aplicativo e reutilize seu conjunto de dados existente
- Use um clone de um conjunto de dados para uma aplicação efêmera
- Reconstruir um cluster do Kubernetes com falha
- Migrar dados da aplicação durante a recuperação de desastre

Considerações

Antes de importar um volume, reveja as seguintes considerações.

- O Trident pode importar apenas volumes ONTAP do tipo RW (leitura-gravação). Os volumes do tipo DP (proteção de dados) são volumes de destino do SnapMirror. Você deve quebrar a relação de espelhamento antes de importar o volume para o Trident.
- Sugerimos importar volumes sem conexões ativas. Para importar um volume usado ativamente, clonar o volume e, em seguida, executar a importação.

  AVISO Isso é especialmente importante para volumes de bloco, já que o Kubernetes não sabia da conexão anterior e poderia facilmente anexar um volume ativo a um pod. Isso pode resultar em corrupção de dados.

- Embora `StorageClass` deva ser especificado em um PVC, o Trident não usa esse parâmetro durante a importação. As classes de armazenamento são usadas durante a criação de volume para selecionar os pools disponíveis com base nas caraterísticas de armazenamento. Como o volume já existe, nenhuma seleção de pool é necessária durante a importação. Portanto, a importação não falhará mesmo se o volume existir em um backend ou pool que não corresponda à classe de armazenamento especificada no PVC.

- O tamanho do volume existente é determinado e definido no PVC. Depois que o volume é importado pelo driver de armazenamento, o PV é criado com uma ClaimRef para o PVC.
  - A política de recuperação é inicialmente definida como `retain` no PV. Depois que o Kubernetes vincula com êxito o PVC e o PV, a política de recuperação é atualizada para corresponder à política de recuperação da Classe de armazenamento.
  - Se a política de recuperação da Storage Class for `delete`, Trident exclui o volume de armazenamento quando o PV é excluído.
- Por padrão, Trident gerencia o PVC e renomeia o FlexVol volume e o LUN ou qtree no backend. Você pode usar a `--no-manage` flag para importar um volume não gerenciado e a `--no-rename` flag para manter o nome do volume.
  - `--no-manage`* - Se você usar o `--no-manage` A Trident não realiza nenhuma operação adicional no PVC ou PV durante o ciclo de vida dos objetos. O volume de armazenamento não é excluído quando o PV é excluído, e outras operações como clonagem e redimensionamento de volume também são ignoradas.
  - `--no-rename`* - Se você usar a `--no-rename`flag, o Trident mantém o nome do volume existente ao importar volumes e gerencia o ciclo de vida dos volumes. Esta opção é compatível apenas com os drivers `ontap-nas`, `ontap-san` (incluindo sistemas ASA r2), `ontap-san-economy` e `ontap-nas-economy` (importação de qtree).

DICA	Essas opções são úteis se você deseja usar o Kubernetes para cargas de trabalho em contêineres, mas, de outra forma, prefere gerenciar o ciclo de vida do volume de armazenamento fora do Kubernetes.

- Uma anotação é adicionada ao PVC e ao PV que serve para um duplo propósito de indicar que o volume foi importado e se o PVC e o PV são gerenciados. Esta anotação não deve ser modificada ou removida.

Importar um volume

Você pode importar um volume usando `tridentctl import` ou criando um PVC com anotações de importação do Trident.

OBSERVAÇÃO	Se você utiliza anotações PVC, não precisa baixar ou usar `tridentctl` para importar o volume.

Usando tridentctl

Passos

1. Crie um arquivo PVC (por exemplo, `pvc.yaml`) que será usado para criar o PVC. O arquivo PVC deve incluir `name`, `namespace`, `accessModes` e `storageClassName`. Opcionalmente, você pode especificar `unixPermissions` na sua definição de PVC.

   O seguinte é um exemplo de uma especificação mínima:

   ```
   kind: PersistentVolumeClaim
   apiVersion: v1
   metadata:
     name: my_claim
     namespace: my_namespace
   ```

```
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: my_storage_class
```

OBSERVAÇÃO	Inclua apenas os parâmetros necessários. Parâmetros adicionais, como o nome do PV ou o tamanho do volume, podem fazer com que o comando de importação falhe.

2. Use o `tridentctl import` Comando para especificar o nome do backend Trident que contém o volume e o nome que identifica exclusivamente o volume no armazenamento (por exemplo: ONTAP FlexVol, Element Volume). O `-f` É necessário fornecer o argumento para especificar o caminho para o arquivo PVC.

```
tridentctl import volume <backendName> <volumeName> -f <path-to-pvc-file>
```

Usando anotações em PVC

Passos

1. Crie um arquivo YAML de PVC (por exemplo, `pvc.yaml`) com as anotações de importação do Trident necessárias. O arquivo de PVC deve incluir:
   - `name` e `namespace` em metadados
   - `accessModes`, `resources.requests.storage`, e `storageClassName` em especificação
   - Anotações:
     - `trident.netapp.io/importOriginalName`: Nome do volume no backend
     - `trident.netapp.io/importBackendUUID`: UUID do backend onde o volume existe
     - `trident.netapp.io/notManaged` (*Opcional*): Defina como `"true"` para volumes não gerenciados. O padrão é `"false"`.

     A seguir está um exemplo de especificação para importar um volume gerenciado:

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: <pvc-name>
  namespace: <namespace>
  annotations:
    trident.netapp.io/importOriginalName: "<volume-name>"
    trident.netapp.io/importBackendUUID: "<backend-uuid>"
spec:
  accessModes:
    - ReadWriteOnce
  resources:
```

```
    requests:
      storage: <size>
  storageClassName: <storage-class-name>
```

2. Aplique o arquivo YAML do PVC ao seu cluster Kubernetes:

```
kubectl apply -f <pvc-file>.yaml
```

Trident importará automaticamente o volume e o vinculará ao PVC.

Exemplos

Reveja os exemplos de importação de volume a seguir para drivers suportados.

ONTAP nas e ONTAP nas FlexGroup

O Trident suporta a importação de volume usando os `ontap-nas` drivers e `ontap-nas-flexgroup`.

OBSERVAÇÃO

- Para importar qtrees com o `ontap-nas-economy` driver, consulte Importar qtrees NFS. Quando o Trident importa uma qtree, ele também importa o FlexVol pai como um bucket que contém a qtree. As qtrees devem ser importadas individualmente.
- Os `ontap-nas` drivers e `ontap-nas-flexgroup` não permitem nomes de volume duplicados.

Cada volume criado com o `ontap-nas` driver é um FlexVol volume no cluster do ONTAP. Importar volumes FlexVol com o `ontap-nas` driver funciona da mesma forma. Um volume FlexVol que já existe em um cluster ONTAP pode ser importado como `ontap-nas` PVC. Da mesma forma, os vols FlexGroup podem ser importados como `ontap-nas-flexgroup` PVCs.

Exemplos de ONTAP NAS usando tridentctl

Os exemplos a seguir mostram como importar volumes gerenciados e não gerenciados usando `tridentctl`.

Volume gerenciado

O exemplo a seguir importa um volume nomeado `managed_volume` em um backend chamado `ontap_nas`:

```
tridentctl import volume ontap_nas managed_volume -f <path-to-pvc-file>

+------------------------------------------+---------+---------------
+----------+------------------------------------------+---------+---------+
|                   NAME                   |  SIZE   | STORAGE CLASS |
PROTOCOL |              BACKEND UUID              | STATE   | MANAGED |
+------------------------------------------+---------+---------------
+----------+------------------------------------------+---------+---------+
```

```
| pvc-bf5ad463-afbb-11e9-8d9f-5254004dfdb7 | 1.0 GiB | standard      |
file     | c5a6f6a4-b052-423b-80d4-8fb491a14a22 | online | true    |
+----------------------------------------+---------+---------------
+---------+----------------------------------------+--------+---------+
```

Volume não gerenciado

Ao usar o `--no-manage` argumento, o Trident não renomeará o volume.

O exemplo a seguir é importado `unmanaged_volume` `ontap_nas` no backend:

```
tridentctl import volume nas_blog unmanaged_volume -f <path-to-pvc-
file> --no-manage

+----------------------------------------+---------+---------------
+---------+----------------------------------------+--------+---------+
|                   NAME                 |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+----------------------------------------+---------+---------------
+---------+----------------------------------------+--------+---------+
| pvc-df07d542-afbc-11e9-8d9f-5254004dfdb7 | 1.0 GiB | standard      |
file     | c5a6f6a4-b052-423b-80d4-8fb491a14a22 | online | false   |
+----------------------------------------+---------+---------------
+---------+----------------------------------------+--------+---------+
```

Exemplos do ONTAP NAS usando anotações PVC

Os exemplos a seguir mostram como importar volumes gerenciados e não gerenciados usando anotações PVC.

Volume gerenciado

O exemplo a seguir importa um volume de 1GiB `ontap-nas` nomeado `ontap_volume1` do backend `81abcb27-ea63-49bb-b606-0a5315ac5f21` com o modo de acesso RWO definido usando anotações PVC:

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: <managed-imported-volume>
  namespace: <namespace>
  annotations:
    trident.netapp.io/importOriginalName: "ontap_volume1"
    trident.netapp.io/importBackendUUID: "81abcb27-ea63-49bb-b606-
0a5315ac5f21"
spec:
```

```
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: <storage-class-name>
```

Volume não gerenciado

O exemplo a seguir importa 1Gi `ontap-nas` volume nomeado `ontap-volume2` do backend `34abcb27-ea63-49bb-b606-0a5315ac5f34` com o modo de acesso RWO definido usando anotações PVC:

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: <umanaged-imported-volume>
  namespace: <namespace>
  annotations:
    trident.netapp.io/importOriginalName: "ontap-volume2"
    trident.netapp.io/importBackendUUID: "34abcb27-ea63-49bb-b606-
0a5315ac5f34"
    trident.netapp.io/notManaged: "true"
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: <storage-class-name>
```

Importar qtrees

OBSERVAÇÃO	A importação de qtree é um recurso em Tech Preview no Trident 26.10. Recursos em Tech Preview não são suportados para uso em produção.

Você pode importar qtrees existentes do ONTAP para o Trident usando o `ontap-nas-economy` driver. Identifique a qtree pelos nomes do volume FlexVol e da qtree no `flexvol/qtree` formato, juntamente com o UUID do backend que a contém. As considerações gerais sobre importação em Visão geral e considerações também se aplicam à importação de qtree.

Importe um qtree usando o tridentctl

Especifique a qtree no formato `flexvol/qtree` como o nome do volume. O exemplo a seguir importa a qtree `qtree1` do volume FlexVol `flexvol1` no backend `ontap_nas_economy`:

```
tridentctl import volume ontap_nas_economy flexvol1/qtree1 -f <path-to-
```

```
pvc-file>
```

Para importar a qtree sem renomeá-la, adicione a `--no-rename` flag. Para importar a qtree sem que Trident gerencie seu ciclo de vida, adicione a `--no-manage` flag.

Importe um qtree usando anotações PVC

Crie um PVC com as seguintes anotações:

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: <pvc-name>
  namespace: <namespace>
  annotations:
    trident.netapp.io/importOriginalName: "flexvol1/qtree1"
    trident.netapp.io/importBackendUUID: "<backend-uuid>"
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: <size>
  storageClassName: <storage-class-name>
```

Para manter os nomes de volume FlexVol e qtree existentes, adicione a `trident.netapp.io/importNoRename: "true"` anotação. Para importar o qtree como não gerenciado, adicione a `trident.netapp.io/notManaged: "true"` anotação.

Comportamento de nomeação e gerenciamento

Importação gerenciada (padrão)

Trident renomeia a qtree para corresponder ao nome do PVC e renomeia o FlexVol volume de acordo com a `ontap-nas-economy` convenção de nomenclatura. O caminho de junção não se altera.

`--no-rename`

Trident mantém os nomes de volume FlexVol e qtree existentes. Os nomes já devem seguir as convenções de nomenclatura do Trident, caso contrário, o Trident não poderá gerenciar operações de exclusão ou montagem do qtree.

`--no-manage`

Trident faz referência à qtree, mas não gerencia seu ciclo de vida. Os nomes devem seguir as convenções de nomenclatura do Trident, e Trident não usa nem gerencia políticas de exportação automática para a qtree.

Comportamento da política de exportação

Se o backend usar políticas de exportação automática, a qtree importada usará o modelo de política de exportação por qtree, no qual cada qtree tem sua própria política de exportação. Trident atribui à qtree uma política de exportação vazia na importação. No momento da publicação, Trident atualiza o volume FlexVol

para a política de exportação por backend se a exportação automática estiver habilitada e a política de exportação FlexVol existente for `default` ou vazia.

CUIDADO	A atualização da política de exportação do volume FlexVol pode interromper outras qtrees no mesmo volume FlexVol que estejam montadas. Importe qtrees somente quando as qtrees no volume FlexVol estiverem desmontadas.

AVISO	Não aponte dois clusters Kubernetes para a mesma SVM ativa e não importe uma qtree que outro cluster ainda gerencia. Se ambos os clusters gerenciarem o mesmo volume, operações conflitantes podem corromper ou destruir os dados.

San ONTAP

O Trident suporta importação de volume usando o `ontap-san` (iSCSI, NVMe/TCP e FC) e `ontap-san-economy` motoristas.

O Trident pode importar volumes ONTAP SAN FlexVol que contêm um único LUN. Isto é consistente com o `ontap-san` driver, que cria um FlexVol volume para cada PVC e um LUN dentro do FlexVol volume. O Trident importa o FlexVol volume e o associa à definição de PVC. Trident pode importar `ontap-san-economy` volumes que contêm vários LUNs.

Os exemplos a seguir mostram como importar volumes gerenciados e não gerenciados:

Volume gerenciado

Para volumes gerenciados, o Trident renomeia o FlexVol volume para `pvc-<uuid>` o formato e o LUN no FlexVol volume para `lun0`.

O exemplo a seguir importa `ontap-san-managed` o FlexVol volume que está presente no `ontap_san_default` back-end:

```
tridentctl import volume ontapsan_san_default ontap-san-managed -f pvc-
basic-import.yaml -n trident -d

+----------------------------------------+--------+--------------
+----------+----------------------------------------+--------+---------+
|                   NAME                 |  SIZE  | STORAGE CLASS |
PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+----------------------------------------+--------+--------------
+----------+----------------------------------------+--------+---------+
| pvc-d6ee4f54-4e40-4454-92fd-d00fc228d74a | 20 MiB | basic         |
block    | cd394786-ddd5-4470-adc3-10c5ce4ca757 | online | true    |
+----------------------------------------+--------+--------------
+----------+----------------------------------------+--------+---------+
```

Volume não gerenciado

O exemplo a seguir é importado `unmanaged_example_volume` `ontap_san` no backend:

```
tridentctl import volume -n trident san_blog unmanaged_example_volume
-f pvc-import.yaml --no-manage
+----------------------------------------+---------+--------------
+---------+----------------------------------------+---------+---------+
|                  NAME                  |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+----------------------------------------+---------+--------------
+---------+----------------------------------------+---------+---------+
| pvc-1fc999c9-ce8c-459c-82e4-ed4380a4b228 | 1.0 GiB | san-blog      |
block    | e3275890-7d80-4af6-90cc-c7a0759f555a | online | false   |
+----------------------------------------+---------+--------------
+---------+----------------------------------------+---------+---------+
```

Se você tiver LUNS mapeados para grupos que compartilham uma IQN com um nó Kubernetes IQN, como mostrado no exemplo a seguir, você receberá o erro: `LUN already mapped to initiator(s) in this group`. Você precisará remover o iniciador ou desmapear o LUN para importar o volume.

```
Vserver   Igroup       Protocol OS Type  Initiators
--------- ------------ -------- -------- ------------------------------------
svm0      k8s-nodename.example.com-fe5d36f2-cded-4f38-9eb0-c7719fc2f9f3
                       iscsi    linux    iqn.1994-05.com.redhat:4c2e1cf35e0
svm0      unmanaged-example-igroup
                       mixed    linux    iqn.1994-05.com.redhat:4c2e1cf35e0
```

Exemplos de SAN-economy do ONTAP

Os exemplos a seguir mostram como importar volumes gerenciados e não gerenciados para o `ontap-san-economy` backend.

Volume gerenciado

Ao importar um volume gerenciado, Trident assume a propriedade do FlexVol e o renomeia. Você deve levar em consideração essa renomeação ao importar vários LUNs do mesmo FlexVol.

O exemplo a seguir importa `lun1` de FlexVol `toimport` como um volume gerenciado chamado `vol-managed-saneco`:

```
tridentctl import volume vol-managed-saneco toimport/lun1 -f
import1.yaml
```

Após a importação `lun1`, Trident renomeia o FlexVol (por exemplo, para `trident_lun_pool_xyz`). Para importar LUNs adicionais do mesmo FlexVol, use o novo nome do FlexVol:

```
tridentctl import volume vol-managed-saneco trident_lun_pool_xyz/lun2
-f import2.yaml
```

OBSERVAÇÃO O `ontap-san-economy` backend importa um LUN por vez. Você pode automatizar várias importações usando um script.

Volume não gerenciado

Ao importar um volume não gerenciado, Trident não assume a propriedade do FlexVol. No entanto, o FlexVol e o LUN devem seguir as convenções de nomenclatura do Trident.

Formato de nomenclatura do FlexVol

```
trident_lun_pool_STORAGEPREFIX_RANDOMSTRING
```

- `STORAGEPREFIX` é o valor de `storagePrefix` na sua configuração de backend. O padrão é `trident`.
- `RANDOMSTRING` é qualquer sequência de caracteres que você escolher.

Requisito de nomenclatura LUN

O LUN deve ser nomeado `lun0`.

Exemplo

Se o seu `storagePrefix` é `xyz`, o caminho completo para o LUN é:

```
trident_lun_pool_xyz_randomstring/lun0
```

Elemento

O Trident suporta o software NetApp Element e a importação de volume NetApp HCI usando o `solidfire-san` driver.

OBSERVAÇÃO O driver Element suporta nomes de volume duplicados. No entanto, o Trident retorna um erro se houver nomes de volume duplicados. Como solução alternativa, clone o volume, forneça um nome de volume exclusivo e importe o volume clonado.

O exemplo a seguir importa um `element-managed` volume no backend . `element_default`

```
tridentctl import volume element_default element-managed -f pvc-basic-
import.yaml -n trident -d
```

```
+-----------------------------------------+--------+--------------
+----------+-----------------------------------------+--------+---------+
|                   NAME                  |  SIZE  | STORAGE CLASS |
PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+-----------------------------------------+--------+--------------
+----------+-----------------------------------------+--------+---------+
| pvc-970ce1ca-2096-4ecd-8545-ac7edc24a8fe | 10 GiB | basic-element |
block    | d3ba047a-ea0b-43f9-9c42-e38e58301c49 | online | true    |
+-----------------------------------------+--------+--------------
+----------+-----------------------------------------+--------+---------+
```

Azure NetApp Files

O Trident suporta a importação de volume usando `azure-netapp-files` o driver.

OBSERVAÇÃO

Para importar um volume Azure NetApp Files, identifique o volume pelo seu caminho de volume. O caminho do volume é a parte do caminho de exportação do volume após o `:/`. Por exemplo, se o caminho de montagem for `10.0.0.2:/importvol1`, o caminho do volume será `importvol1`.

O exemplo a seguir importa um `azure-netapp-files` volume no back-end `azurenetappfiles_40517` com o caminho do volume `importvol1` .

```
tridentctl import volume azurenetappfiles_40517 importvol1 -f <path-to-
pvc-file> -n trident

+-----------------------------------------+---------+--------------
+----------+-----------------------------------------+--------+---------+
|                   NAME                  |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+-----------------------------------------+---------+--------------
+----------+-----------------------------------------+--------+---------+
| pvc-0ee95d60-fd5c-448d-b505-b72901b3a4ab | 100 GiB | anf-storage   |
file     | 1c01274f-d94b-44a3-98a3-04c953c9a51e | online | true    |
+-----------------------------------------+---------+--------------
+----------+-----------------------------------------+--------+---------+
```

Google Cloud NetApp volumes

O Trident suporta a importação de volume usando `google-cloud-netapp-volumes` o driver.

O exemplo a seguir importa um volume no backend `backend-tbc-gcnv1` com o volume `testvoleasiaeast1`.

```
tridentctl import volume backend-tbc-gcnv1 "testvoleasiaeast1" -f < path-
to-pvc> -n trident
```

```
+----------------------------------------+---------
+---------------------+----------+--------------------------------------
+--------+---------+
|                   NAME                   |  SIZE   | STORAGE CLASS
| PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+----------------------------------------+---------
+---------------------+----------+--------------------------------------
+--------+---------+
| pvc-a69cda19-218c-4ca9-a941-aea05dd13dc0 |  10 GiB | gcnv-nfs-sc-
identity | file     | 8c18cdf1-0770-4bc0-bcc5-c6295fe6d837 | online | true
|
+----------------------------------------+---------
+---------------------+----------+--------------------------------------
+--------+---------+
```

O exemplo a seguir importa um `google-cloud-netapp-volumes` volume quando dois volumes estão presentes na mesma região:

```
tridentctl import volume backend-tbc-gcnv1
"projects/123456789100/locations/asia-east1-a/volumes/testvoleasiaeast1"
-f <path-to-pvc> -n trident

+----------------------------------------+---------
+---------------------+----------+--------------------------------------
+--------+---------+
|                   NAME                   |  SIZE   | STORAGE CLASS
| PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+----------------------------------------+---------
+---------------------+----------+--------------------------------------
+--------+---------+
| pvc-a69cda19-218c-4ca9-a941-aea05dd13dc0 |  10 GiB | gcnv-nfs-sc-
identity | file     | 8c18cdf1-0770-4bc0-bcc5-c6295fe6d837 | online | true
|
+----------------------------------------+---------
+---------------------+----------+--------------------------------------
+--------+---------+
```

Personalize os nomes e rótulos de volume do Trident

Com o Trident, você pode atribuir nomes e rótulos significativos aos volumes criados. Isso ajuda a identificar e mapear facilmente volumes para seus respectivos recursos do Kubernetes (PVCs). Você também pode definir modelos no nível de back-end para criar nomes de volume personalizados e rótulos personalizados; todos os volumes que você criar, importar ou clonar aderirão aos modelos.

Antes de começar

Nomes de volume e etiquetas personalizáveis suportam:

- Operações de criação, importação e clonagem de volume.
- No caso do `ontap-nas-economy` No driver, apenas o nome do volume Qtree está em conformidade com o modelo de nome.
- No caso do `ontap-san-economy` No driver, apenas o nome da LUN está em conformidade com o modelo de nome.

Limitações

- Os nomes de volumes personalizados são compatíveis apenas com drivers ONTAP locais.
- Etiquetas personalizadas são suportadas apenas para o `ontap-san`, `ontap-nas`, e `ontap-nas-flexgroup` motoristas.
- Nomes de volume personalizados não se aplicam a volumes existentes.

Principais comportamentos de nomes de volume personalizáveis

- Se ocorrer uma falha devido à sintaxe inválida em um modelo de nome, a criação do backend falhará. No entanto, se o aplicativo modelo falhar, o volume será nomeado de acordo com a convenção de nomenclatura existente.
- O prefixo de armazenamento não é aplicável quando um volume é nomeado usando um modelo de nome da configuração de back-end. Qualquer valor de prefixo desejado pode ser adicionado diretamente ao modelo.

Exemplos de configuração de backend com modelo de nome e rótulos

Modelos de nome personalizados podem ser definidos no nível raiz e/ou pool.

Exemplo de nível de raiz

```
{
  "version": 1,
  "storageDriverName": "ontap-nas",
  "backendName": "ontap-nfs-backend",
  "managementLIF": "<ip address>",
  "svm": "svm0",
  "username": "<admin>",
  "password": "<password>",
  "defaults": {
    "nameTemplate":
"{{.volume.Name}}_{{.labels.cluster}}_{{.volume.Namespace}}_{{.volume.Requ
estName}}"
  },
  "labels": {
    "cluster": "ClusterA",
    "PVC": "{{.volume.Namespace}}_{{.volume.RequestName}}"
  }
```

```
}
```

Exemplo de nível de pool

```
{
  "version": 1,
  "storageDriverName": "ontap-nas",
  "backendName": "ontap-nfs-backend",
  "managementLIF": "<ip address>",
  "svm": "svm0",
  "username": "<admin>",
  "password": "<password>",
  "useREST": true,
  "storage": [
    {
      "labels": {
        "labelname": "label1",
        "name": "{{ .volume.Name }}"
      },
      "defaults": {
        "nameTemplate": "pool01_{{ .volume.Name }}_{{ .labels.cluster
}}_{{ .volume.Namespace }}_{{ .volume.RequestName }}"
      }
    },
    {
      "labels": {
        "cluster": "label2",
        "name": "{{ .volume.Name }}"
      },
      "defaults": {
        "nameTemplate": "pool02_{{ .volume.Name }}_{{ .labels.cluster
}}_{{ .volume.Namespace }}_{{ .volume.RequestName }}"
      }
    }
  ]
}
```

Exemplos de modelo de nome

Exemplo 1:

```
"nameTemplate": "{{ .config.StoragePrefix }}_{{ .volume.Name }}_{{
.config.BackendName }}"
```

Exemplo 2:

```
"nameTemplate": "pool_{{ .config.StoragePrefix }}_{{ .volume.Name }}_{{
slice .volume.RequestName 1 5 }}""
```

Pontos a considerar

1. No caso das importações de volume, as etiquetas são atualizadas apenas se o volume existente tiver etiquetas num formato específico. Por exemplo `{"provisioning":{"Cluster":"ClusterA", "PVC": "pvcname"}}`.
2. No caso de importações de volume gerenciado, o nome do volume segue o modelo de nome definido no nível raiz na definição de back-end.
3. O Trident não suporta a utilização de um operador de corte com o prefixo de armazenamento.
4. Se os modelos não resultarem em nomes de volume exclusivos, o Trident adicionará alguns carateres aleatórios para criar nomes de volume exclusivos.
5. Se o nome personalizado para um volume de economia nas exceder 64 carateres de comprimento, o Trident nomeará os volumes de acordo com a convenção de nomenclatura existente. Para todos os outros drivers ONTAP, se o nome do volume exceder o limite de nome, o processo de criação de volume falhará.

Compartilhe um volume NFS do Trident entre namespaces

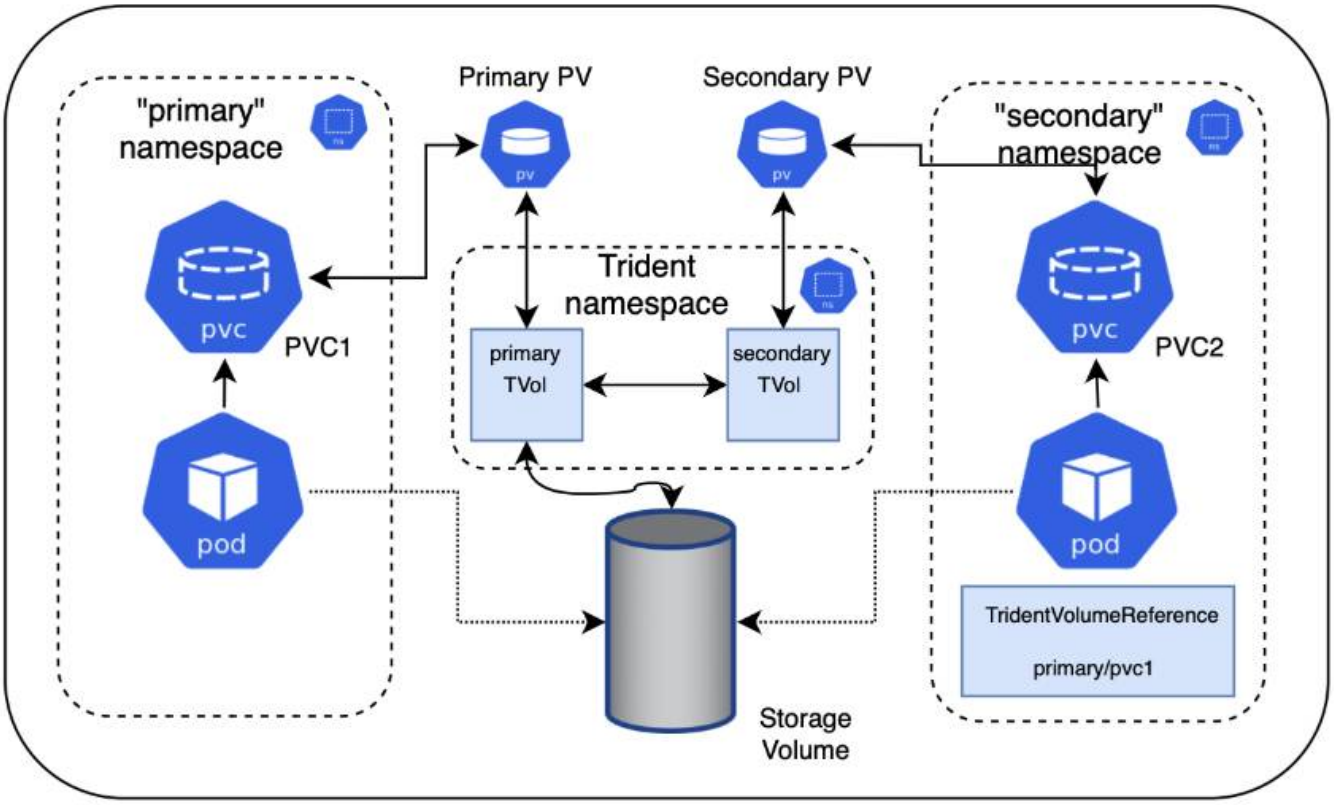
Usando o Trident, você pode criar um volume em um namespace principal e compartilhá-lo em um ou mais namespaces secundários.

Caraterísticas

O TridentVolumeReference CR permite que você compartilhe com segurança volumes NFS ReadWriteMany (RWX) em um ou mais namespaces do Kubernetes. Essa solução nativa do Kubernetes tem os seguintes benefícios:

- Vários níveis de controle de acesso para garantir a segurança
- Funciona com todos os drivers de volume Trident NFS
- Não há dependência do tridentctl ou de qualquer outro recurso do Kubernetes não nativo

Este diagrama ilustra o compartilhamento de volumes NFS em dois namespaces do Kubernetes.



Início rápido

Você pode configurar o compartilhamento de volume NFS em apenas algumas etapas.

1 **Configure o PVC de origem para compartilhar o volume**

O proprietário do namespace de origem concede permissão para acessar os dados no PVC de origem.

2 **Conceda permissão para criar um CR no namespace de destino**

O administrador do cluster concede permissão ao proprietário do namespace de destino para criar o CredentVolumeReference CR.

3 **Crie TridentVolumeReference no namespace de destino**

O proprietário do namespace de destino cria o TrigentVolumeReference CR para se referir ao PVC de origem.

4 **Crie o PVC subordinado no namespace de destino**

O proprietário do namespace de destino cria o PVC subordinado para usar a fonte de dados do PVC de origem.

Configure os namespaces de origem e destino

Para garantir a segurança, o compartilhamento entre namespace requer colaboração e ação do proprietário

do namespace de origem, do administrador do cluster e do proprietário do namespace de destino. A função de usuário é designada em cada etapa.

Passos

1. **Proprietário do namespace de origem:** Crie o PVC (`pvc1`) no namespace de origem que concede permissão para compartilhar com o namespace de destino (`namespace2`) usando a `shareToNamespace` anotação.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc1
  namespace: namespace1
  annotations:
    trident.netapp.io/shareToNamespace: namespace2
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: trident-csi
  resources:
    requests:
      storage: 100Gi
```

   A Trident cria o PV e seu volume de storage NFS no back-end.

   OBSERVAÇÃO

   - Você pode compartilhar o PVC para vários namespaces usando uma lista delimitada por vírgulas. Por exemplo, `trident.netapp.io/shareToNamespace: namespace2,namespace3,namespace4`.
   - Você pode compartilhar com todos os namespaces usando `*`. Por exemplo, `trident.netapp.io/shareToNamespace: *`
   - Você pode atualizar o PVC para incluir a `shareToNamespace` anotação a qualquer momento.

2. **Administrador do cluster:** Certifique-se de que o RBAC adequado esteja em vigor para conceder permissão ao proprietário do namespace de destino para criar o TridentVolumeReference CR no namespace de destino.
3. **Proprietário do namespace de destino:** Crie um CredentVolumeReference CR no namespace de destino que se refere ao namespace de origem `pvc1` .

```
apiVersion: trident.netapp.io/v1
kind: TridentVolumeReference
metadata:
  name: my-first-tvr
  namespace: namespace2
spec:
```

```
  pvcName: pvc1
  pvcNamespace: namespace1
```

4. **Proprietário do namespace de destino:** Crie um PVC (`pvc2`) no namespace de destino (`namespace2`) usando a `shareFromPVC` anotação para designar o PVC de origem.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  annotations:
    trident.netapp.io/shareFromPVC: namespace1/pvc1
  name: pvc2
  namespace: namespace2
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: trident-csi
  resources:
    requests:
      storage: 100Gi
```

OBSERVAÇÃO	O tamanho do PVC de destino deve ser inferior ou igual ao PVC de origem.

Resultados

O Trident lê a `shareFromPVC` anotação no PVC de destino e cria o PV de destino como um volume subordinado sem recurso de armazenamento próprio que aponta para o PV de origem e compartilha o recurso de armazenamento PV de origem. O PVC e o PV de destino aparecem encadernados normalmente.

Eliminar um volume partilhado

Você pode excluir um volume compartilhado entre vários namespaces. O Trident removerá o acesso ao volume no namespace de origem e manterá o acesso para outros namespaces que compartilham o volume. Quando todos os namespaces que fazem referência ao volume são removidos, o Trident exclui o volume.

`tridentctl get`Use para consultar volumes subordinados

Usando o[`tridentctl` utilitário, você pode executar o `get` comando para obter volumes subordinados. Para mais informações, consulte `tridentctl` comandos e opções.

```
Usage:
  tridentctl get [option]
```

Bandeiras -

- `-h, --help`: Ajuda para volumes.

- `--parentOfSubordinate string`: Limitar consulta ao volume de origem subordinado.
- `--subordinateOf string`: Limitar consulta a subordinados de volume.

Limitações

- O Trident não pode impedir que namespaces de destino gravem no volume compartilhado. Você deve usar o bloqueio de arquivos ou outros processos para evitar a substituição de dados de volume compartilhado.
- Não é possível revogar o acesso ao PVC de origem removendo as `shareToNamespace` anotações ou ou `shareFromNamespace` excluindo o `TridentVolumeReference` CR. Para revogar o acesso, você deve excluir o PVC subordinado.
- Snapshots, clones e espelhamento não são possíveis em volumes subordinados.

Para mais informações

Para saber mais sobre o acesso ao volume entre namespace:

- Assista à demonstração no "NetAppTV".

Clone volumes do Trident em namespaces diferentes

Com o Trident, você pode criar novos volumes usando volumes existentes ou volumes de volume a partir de um namespace diferente dentro do mesmo cluster do Kubernetes.

Pré-requisitos

Antes de clonar volumes, certifique-se de que os backends de origem e destino são do mesmo tipo e têm a mesma classe de armazenamento.

OBSERVAÇÃO	A clonagem entre namespaces é suportada apenas para `ontap-san` e `ontap-nas` drivers de armazenamento. Clones somente leitura não são suportados.

Início rápido

Você pode configurar a clonagem de volume em apenas algumas etapas.



Configure o PVC de origem para clonar o volume

O proprietário do namespace de origem concede permissão para acessar os dados no PVC de origem.



Conceda permissão para criar um CR no namespace de destino

O administrador do cluster concede permissão ao proprietário do namespace de destino para criar o CredentVolumeReference CR.



Crie TridentVolumeReference no namespace de destino

O proprietário do namespace de destino cria o TrigentVolumeReference CR para se referir ao PVC de origem.

4 Crie o clone PVC no namespace de destino

O proprietário do namespace de destino cria PVC para clonar o PVC a partir do namespace de origem.

Configure os namespaces de origem e destino

Para garantir a segurança, a clonagem de volumes entre namespaces requer colaboração e ação do proprietário do namespace de origem, do administrador do cluster e do proprietário do namespace de destino. A função de usuário é designada em cada etapa.

Passos

1. **Proprietário do namespace de origem:** Crie o PVC (`pvc1`) no namespace de origem (`namespace1`) que concede permissão para compartilhar com o namespace de destino (`namespace2`) usando a `cloneToNamespace` anotação.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc1
  namespace: namespace1
  annotations:
    trident.netapp.io/cloneToNamespace: namespace2
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: trident-csi
  resources:
    requests:
      storage: 100Gi
```

   O Trident cria o PV e seu volume de armazenamento de back-end.

   OBSERVAÇÃO

   - Você pode compartilhar o PVC para vários namespaces usando uma lista delimitada por vírgulas. Por exemplo, `trident.netapp.io/cloneToNamespace: namespace2,namespace3,namespace4`.
   - Você pode compartilhar com todos os namespaces usando `*`. Por exemplo, `trident.netapp.io/cloneToNamespace: *`
   - Você pode atualizar o PVC para incluir a `cloneToNamespace` anotação a qualquer momento.

2. **Administrador do cluster:** Certifique-se de que o RBAC adequado esteja em vigor para conceder permissão ao proprietário do namespace de destino para criar o CR TridentVolumeReference no namespace de destino(`namespace2`).
3. **Proprietário do namespace de destino:** crie um TridentVolumeReference CR no namespace de destino que faça referência ao PVC de origem `pvc1`.

```
apiVersion: trident.netapp.io/v1
kind: TridentVolumeReference
metadata:
  name: my-first-tvr
  namespace: namespace2
spec:
  pvcName: pvc1
  pvcNamespace: namespace1
```

4. **Proprietário do namespace de destino:** Crie um PVC (`pvc2`) no namespace de destino (`namespace2`) usando `cloneFromPVC` ou `cloneFromSnapshot` e `cloneFromNamespace` anotações para designar o PVC de origem.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  annotations:
    trident.netapp.io/cloneFromPVC: pvc1
    trident.netapp.io/cloneFromNamespace: namespace1
  name: pvc2
  namespace: namespace2
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: trident-csi
  resources:
    requests:
      storage: 100Gi
```

Limitações

- Para PVCs provisionados com drivers de economia ONTAP-nas, os clones somente leitura não são compatíveis.

Replique volumes usando o SnapMirror

O Trident oferece suporte a relacionamentos espelhados entre um volume de origem em um cluster e o volume de destino no cluster com peering para replicação de dados para recuperação de desastres. Você pode usar uma Definição de Recurso Personalizada (CRD) com namespace, chamada Trident Mirror Relationship (TMR), para executar as seguintes operações:

- Criar relações de espelhamento entre volumes (PVCs)
- Remova as relações de espelho entre volumes

- Quebre as relações do espelho
- Promover o volume secundário durante as condições de desastre (failovers)
- Realizar a transição sem perda de aplicativos do cluster para o cluster (durante failovers planejados ou migrações)

Pré-requisitos de replicação

Certifique-se de que os seguintes pré-requisitos sejam atendidos antes de começar:

Clusters de ONTAP

- **Trident**: O Trident versão 22,10 ou posterior deve existir nos clusters do Kubernetes de origem e destino que utilizam o ONTAP como um back-end.
- **Licenças**: As licenças assíncronas do ONTAP SnapMirror usando o pacote proteção de dados devem estar ativadas nos clusters ONTAP de origem e destino. "Visão geral do licenciamento do SnapMirror no ONTAP"Consulte para obter mais informações.

  A partir do ONTAP 9.10,1, todas as licenças são entregues como um arquivo de licença NetApp (NLF), que é um único arquivo que permite vários recursos. "Licenças incluídas no ONTAP One"Consulte para obter mais informações.

OBSERVAÇÃO	Somente a proteção assíncrona SnapMirror é suportada.

Peering

- **Cluster e SVM**: Os backends de storage do ONTAP devem ser colocados em Contato. "Visão geral do peering de cluster e SVM"Consulte para obter mais informações.

IMPORTANTE	Certifique-se de que os nomes do SVM usados na relação de replicação entre dois clusters ONTAP sejam exclusivos.

- **Trident e SVM**: Os SVMs remotas com peering devem estar disponíveis para o Trident no cluster de destino.

Drivers suportados

O NetApp Trident oferece suporte à replicação de volume com a tecnologia NetApp SnapMirror usando classes de armazenamento apoiadas pelos seguintes drivers: **`ontap-nas` : NFS** `ontap-san` : iSCSI **`ontap-san` :FC** `ontap-san` : NVMe/TCP (requer no mínimo a versão ONTAP 9.15.1)

OBSERVAÇÃO	A replicação de volume usando SnapMirror não é compatível com sistemas ASA r2. Para obter informações sobre sistemas ASA r2, consulte "Saiba mais sobre os sistemas de armazenamento ASA R2" .

Crie um PVC espelhado

Siga estas etapas e use os exemplos CRD para criar relação de espelhamento entre volumes primário e secundário.

Passos

1. Execute as etapas a seguir no cluster primário do Kubernetes:
   a. Crie um objeto StorageClass com o `trident.netapp.io/replication: true` parâmetro.

Exemplo

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: csi-nas
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-nas"
  fsType: "nfs"
  trident.netapp.io/replication: "true"
```

b. Crie um PVC com StorageClass criado anteriormente.

Exemplo

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: csi-nas
spec:
  accessModes:
  - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: csi-nas
```

c. Crie um MirrorRelationship CR com informações locais.

Exemplo

```
kind: TridentMirrorRelationship
apiVersion: trident.netapp.io/v1
metadata:
  name: csi-nas
spec:
  state: promoted
  volumeMappings:
  - localPVCName: csi-nas
```

O Trident obtém as informações internas para o volume e o estado atual de proteção de dados (DP) do volume e, em seguida, preenche o campo de status do espelhamento.

d. Obtenha o tridentMirrorRelacionship CR para obter o nome interno e SVM do PVC.

```
kubectl get tmr csi-nas
```

```
kind: TridentMirrorRelationship
apiVersion: trident.netapp.io/v1
metadata:
  name: csi-nas
  generation: 1
spec:
  state: promoted
  volumeMappings:
  - localPVCName: csi-nas
status:
  conditions:
  - state: promoted
    localVolumeHandle:
"datavserver:trident_pvc_3bedd23c_46a8_4384_b12b_3c38b313c1e1"
    localPVCName: csi-nas
    observedGeneration: 1
```

2. Execute as etapas a seguir no cluster secundário do Kubernetes:
   a. Crie um StorageClass com o parâmetro Trident.NetApp.io/replicação: True.

      Exemplo

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: csi-nas
provisioner: csi.trident.netapp.io
parameters:
  trident.netapp.io/replication: true
```

   b. Crie um MirrorRelationship CR com informações de destino e origem.

      Exemplo

```
kind: TridentMirrorRelationship
apiVersion: trident.netapp.io/v1
metadata:
  name: csi-nas
spec:
  state: established
  replicationPolicy: MirrorAllSnapshots
  replicationSchedule: 5min
```

```
  volumeMappings:
  - localPVCName: csi-nas
    remoteVolumeHandle:
"datavserver:trident_pvc_3bedd23c_46a8_4384_b12b_3c38b313c1e1"
```

O TMR `spec` possui quatro campos de nível superior: `state`, que define o estado de relacionamento desejado (consulte Estados de replicação de volume); `volumeMappings`, que associa PVCs locais aos seus identificadores de volume remotos; e os campos opcionais `replicationPolicy` e `replicationSchedule` descritos abaixo.

Os campos `replicationPolicy` e `replicationSchedule` são opcionais. Defina `replicationPolicy` como o nome de uma política assíncrona do SnapMirror no ONTAP e `replicationSchedule` como o nome de um agendamento do ONTAP. Se um dos campos for omitido, o Trident usará a política ou o agendamento de replicação correspondente configurado no backend. Defina esses campos ao criar a relação; você não poderá alterar a política ou o agendamento depois que a relação for estabelecida. Para usar configurações diferentes, exclua e recrie o TMR.

O Trident cria e inicializa uma relação do SnapMirror usando a política e o agendamento de replicação configurados.

1. Crie um PVC com StorageClass criado anteriormente para atuar como secundário (destino SnapMirror).

   Exemplo

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: csi-nas
  annotations:
    trident.netapp.io/mirrorRelationship: csi-nas
spec:
  accessModes:
  - ReadWriteMany
resources:
  requests:
    storage: 1Gi
storageClassName: csi-nas
```

   O Trident verificará se há CRD de relacionamento de espelhamento e não criará o volume se a relação não existir. Se o relacionamento existir, o Trident garantirá que o novo FlexVol volume seja colocado em um SVM que seja emparelhado com o SVM remoto definido no espelhamento.

Estados de replicação de volume

Um relacionamento de espelhamento do Trident (TMR) é um CRD que representa um fim de uma relação de replicação entre PVCs. O TMR de destino tem um estado, que informa ao Trident qual é o estado desejado. O TMR de destino tem os seguintes estados:

- * Estabelecido*: O PVC local é o volume de destino de uma relação de espelho, e esta é uma nova relação.

- **Promovido**: O PVC local é ReadWrite e montável, sem relação de espelho atualmente em vigor.
- * Restabelecido*: O PVC local é o volume de destino de uma relação de espelho e também estava anteriormente nessa relação de espelho.
  - O estado restabelecido deve ser usado se o volume de destino estiver em uma relação com o volume de origem, porque ele sobrescreve o conteúdo do volume de destino.
  - O estado restabelecido falhará se o volume não estiver previamente em uma relação com a fonte.

Promover PVC secundário durante um failover não planejado

Execute a seguinte etapa no cluster secundário do Kubernetes:

- Atualize o campo *spec.State* do TrigentMirrorRelationship para `promoted`.

Promover PVC secundário durante um failover planejado

Durante um failover planejado (migração), execute as seguintes etapas para promover o PVC secundário:

Passos

1. No cluster primário do Kubernetes, crie um snapshot do PVC e aguarde até que o snapshot seja criado.
2. No cluster Kubernetes primário, crie o TridentTridentSnapshotInfoCR para obter detalhes internos.

   Exemplo

   ```
   kind: TridentSnapshotInfo
   apiVersion: trident.netapp.io/v1
   metadata:
     name: csi-nas
   spec:
     snapshotName csi-nas-snapshot
   ```

   a. No cluster Kubernetes secundário, atualize o campo *spec.state* do CR *TridentMirrorRelationship* para *promoted* e *spec.volumeMappings[].promotedSnapshotHandle* para ser o internalName do snapshot.
3. No cluster secundário do Kubernetes, confirme o status (campo status.State) do TrigentMirrorRelationship para promovido.

Restaurar uma relação de espelhamento após um failover

Antes de restaurar uma relação de espelho, escolha o lado que você deseja fazer como o novo primário.

Passos

1. No cluster secundário do Kubernetes, certifique-se de que os valores do campo *spec.remoteVolumeHandle* no TrigentMirrorRelationship sejam atualizados.
2. No cluster Kubernetes secundário, atualize o campo *spec.state* de TridentMirrorRelationship para `reestablished`.

Operações adicionais

O Trident dá suporte às seguintes operações nos volumes primário e secundário:

Replique PVC primário para um novo PVC secundário

Certifique-se de que você já tem um PVC primário e um PVC secundário.

Passos

1. Exclua as CRDs PersistentVolumeClaim e TridentMirrorRelacionship do cluster secundário (destino) estabelecido.
2. Exclua o CRD do tridentMirrorRelationship do cluster primário (de origem).
3. Crie um novo CRD de TridentMirrorRelacionship no cluster primário (de origem) para o novo PVC secundário (de destino) que você deseja estabelecer.

Redimensione um PVC espelhado, primário ou secundário

O PVC pode ser redimensionado como normal, o ONTAP irá expandir automaticamente qualquer destino flevxols se a quantidade de dados exceder o tamanho atual.

Remova a replicação de um PVC

Para remover a replicação, execute uma das seguintes operações no volume secundário atual:

- Exclua o MirrorRelationship no PVC secundário. Isso quebra a relação de replicação.
- Ou atualize o campo spec.State para *promovido*.

Excluir um PVC (que foi anteriormente espelhado)

O Trident verifica se há PVCs replicados e libera a relação de replicação antes de tentar excluir o volume.

Eliminar um TMR

A exclusão de um TMR em um lado de um relacionamento espelhado faz com que o TMR restante passe para o estado *promovido* antes que o Trident conclua a exclusão. Se o TMR selecionado para exclusão já estiver no estado *promovido*, não há relacionamento de espelhamento existente e o TMR será removido e o Trident promoverá o PVC local para *ReadWrite*. Essa exclusão libera os metadados do SnapMirror para o volume local no ONTAP. Se este volume for usado em uma relação de espelho no futuro, ele deve usar um novo TMR com um estado de replicação de volume *established* ao criar a nova relação de espelho.

Atualizar relações de espelho quando o ONTAP estiver online

As relações de espelho podem ser atualizadas a qualquer momento depois que são estabelecidas. Pode utilizar os `state: promoted` campos ou `state: reestablished` para atualizar as relações. Ao promover um volume de destino para um volume ReadWrite regular, você pode usar *promotedSnapshotHandle* para especificar um snapshot específico para restaurar o volume atual.

Atualizar relações de espelho quando o ONTAP estiver offline

Você pode usar um CRD para executar uma atualização do SnapMirror sem que o Trident tenha conetividade direta com o cluster do ONTAP. Consulte o seguinte formato de exemplo do TrigentActionMirrorUpdate:

Exemplo

```
apiVersion: trident.netapp.io/v1
kind: TridentActionMirrorUpdate
metadata:
```

```
  name: update-mirror-b
spec:
  snapshotHandle: "pvc-1234/snapshot-1234"
  tridentMirrorRelationshipName: mirror-b
```

`status.state` Reflete o estado do CRD do TrigentActionMirrorUpdate. Ele pode tomar um valor de *successful*, *in progress* ou *Failed*.

Use a topologia CSI

O Trident pode criar e anexar volumes seletivamente a nós presentes em um cluster do Kubernetes, utilizando o "Recurso de topologia CSI".

Visão geral

Usando o recurso de topologia de CSI, o acesso a volumes pode ser limitado a um subconjunto de nós, com base em regiões e zonas de disponibilidade. Hoje em dia, os provedores de nuvem permitem que os administradores do Kubernetes gerem nós baseados em zonas. Os nós podem ser localizados em diferentes zonas de disponibilidade dentro de uma região ou em várias regiões. Para facilitar o provisionamento de volumes para workloads em uma arquitetura de várias zonas, o Trident usa a topologia de CSI.

DICA Saiba mais sobre o recurso de topologia de CSI "aqui" .

O Kubernetes oferece dois modos exclusivos de vinculação de volume:

- Com `VolumeBindingMode` definido como `Immediate`, o Trident cria o volume sem qualquer reconhecimento de topologia. A vinculação de volume e o provisionamento dinâmico são tratados quando o PVC é criado. Esse é o padrão `VolumeBindingMode` e é adequado para clusters que não impõem restrições de topologia. Os volumes persistentes são criados sem depender dos requisitos de agendamento do pod solicitante.
- Com `VolumeBindingMode` definido como `WaitForFirstConsumer`, a criação e a vinculação de um volume persistente para um PVC é adiada até que um pod que usa o PVC seja programado e criado. Dessa forma, os volumes são criados para atender às restrições de agendamento impostas pelos requisitos de topologia.

OBSERVAÇÃO O `WaitForFirstConsumer` modo de encadernação não requer rótulos de topologia. Isso pode ser usado independentemente do recurso de topologia de CSI.

O que você vai precisar

Para fazer uso da topologia de CSI, você precisa do seguinte:

- Um cluster de Kubernetes executando um "Versão do Kubernetes compatível"

```
kubectl version
Client Version: version.Info{Major:"1", Minor:"19",
GitVersion:"v1.19.3",
GitCommit:"1e11e4a2108024935ecfcb2912226cedeafd99df",
GitTreeState:"clean", BuildDate:"2020-10-14T12:50:19Z",
```

```
GoVersion:"go1.15.2", Compiler:"gc", Platform:"linux/amd64"}
Server Version: version.Info{Major:"1", Minor:"19",
GitVersion:"v1.19.3",
GitCommit:"1e11e4a2108024935ecfcb2912226cedeafd99df",
GitTreeState:"clean", BuildDate:"2020-10-14T12:41:49Z",
GoVersion:"go1.15.2", Compiler:"gc", Platform:"linux/amd64"}
```

- Os nós no cluster devem ter rótulos que introduzam consciência de topologia `topology.kubernetes.io/region` e `topology.kubernetes.io/zone`). O `topology.kubernetes.io/region` rótulo é obrigatório para o provisionamento com reconhecimento de topologia no Trident. Esses rótulos devem estar presentes nos nós antes da instalação do Trident. Se você adicionar ou alterar rótulos de topologia após a instalação, reiniciar os pods de nó do Trident é suficiente para que o Trident reconheça as alterações.

```
kubectl get nodes -o=jsonpath='{range .items[*]}[{.metadata.name},
{.metadata.labels}]{"\n"}{end}' | grep --color "topology.kubernetes.io"
[node1,
{"beta.kubernetes.io/arch":"amd64","beta.kubernetes.io/os":"linux","kube
rnetes.io/arch":"amd64","kubernetes.io/hostname":"node1","kubernetes.io/
os":"linux","node-
role.kubernetes.io/master":"","topology.kubernetes.io/region":"us-
east1","topology.kubernetes.io/zone":"us-east1-a"}]
[node2,
{"beta.kubernetes.io/arch":"amd64","beta.kubernetes.io/os":"linux","kube
rnetes.io/arch":"amd64","kubernetes.io/hostname":"node2","kubernetes.io/
os":"linux","node-
role.kubernetes.io/worker":"","topology.kubernetes.io/region":"us-
east1","topology.kubernetes.io/zone":"us-east1-b"}]
[node3,
{"beta.kubernetes.io/arch":"amd64","beta.kubernetes.io/os":"linux","kube
rnetes.io/arch":"amd64","kubernetes.io/hostname":"node3","kubernetes.io/
os":"linux","node-
role.kubernetes.io/worker":"","topology.kubernetes.io/region":"us-
east1","topology.kubernetes.io/zone":"us-east1-c"}]
```

Etapa 1: Crie um back-end com reconhecimento de topologia

Os back-ends de storage do Trident podem ser projetados para provisionar volumes seletivamente de acordo com as zonas de disponibilidade. Cada back-end pode transportar um bloco opcional `supportedTopologies` que representa uma lista de zonas e regiões com suporte. Para o StorageClasses que fazem uso de tal back-end, um volume só seria criado se solicitado por um aplicativo agendado em uma região/zona suportada.

Aqui está um exemplo de definição de backend:

YAML

```yaml
---
version: 1
storageDriverName: ontap-san
backendName: san-backend-us-east1
managementLIF: 192.168.27.5
svm: iscsi_svm
username: admin
password: password
supportedTopologies:
  - topology.kubernetes.io/region: us-east1
    topology.kubernetes.io/zone: us-east1-a
  - topology.kubernetes.io/region: us-east1
    topology.kubernetes.io/zone: us-east1-b
```

JSON

```json
{
  "version": 1,
  "storageDriverName": "ontap-san",
  "backendName": "san-backend-us-east1",
  "managementLIF": "192.168.27.5",
  "svm": "iscsi_svm",
  "username": "admin",
  "password": "password",
  "supportedTopologies": [
    {
      "topology.kubernetes.io/region": "us-east1",
      "topology.kubernetes.io/zone": "us-east1-a"
    },
    {
      "topology.kubernetes.io/region": "us-east1",
      "topology.kubernetes.io/zone": "us-east1-b"
    }
  ]
}
```

OBSERVAÇÃO

`supportedTopologies` é usado para fornecer uma lista de regiões e zonas por backend. Essas regiões e zonas representam a lista de valores permitidos que podem ser fornecidos em um StorageClass. Para os StorageClasses que contêm um subconjunto das regiões e zonas fornecidas em um back-end, o Trident cria um volume no back-end.

Você também pode definir `supportedTopologies` por pool de armazenamento. Veja o exemplo a seguir:

```
---
version: 1
storageDriverName: ontap-nas
backendName: nas-backend-us-central1
managementLIF: 172.16.238.5
svm: nfs_svm
username: admin
password: password
supportedTopologies:
  - topology.kubernetes.io/region: us-central1
    topology.kubernetes.io/zone: us-central1-a
  - topology.kubernetes.io/region: us-central1
    topology.kubernetes.io/zone: us-central1-b
storage:
  - labels:
      workload: production
    supportedTopologies:
      - topology.kubernetes.io/region: us-central1
        topology.kubernetes.io/zone: us-central1-a
  - labels:
      workload: dev
    supportedTopologies:
      - topology.kubernetes.io/region: us-central1
        topology.kubernetes.io/zone: us-central1-b
```

Neste exemplo, as `region` etiquetas e `zone` representam a localização do conjunto de armazenamento. `topology.kubernetes.io/region` `topology.kubernetes.io/zone` e dite de onde os pools de storage podem ser consumidos.

Etapa 2: Defina StorageClasses que estejam cientes da topologia

Com base nas etiquetas de topologia fornecidas aos nós no cluster, o StorageClasses pode ser definido para conter informações de topologia. Isso determinará os pools de storage que atuam como candidatos a solicitações de PVC feitas e o subconjunto de nós que podem fazer uso dos volumes provisionados pelo Trident.

Veja o exemplo a seguir:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: netapp-san-us-east1
provisioner: csi.trident.netapp.io
volumeBindingMode: WaitForFirstConsumer
```

```
allowedTopologies:
  - matchLabelExpressions:
    - key: topology.kubernetes.io/zone
      values:
        - us-east1-a
        - us-east1-b
    - key: topology.kubernetes.io/region
      values:
        - us-east1
parameters:
  fsType: ext4
```

Na definição StorageClass fornecida acima, `volumeBindingMode` está definida como `WaitForFirstConsumer`. Os PVCs solicitados com este StorageClass não serão utilizados até que sejam referenciados em um pod. E, `allowedTopologies` fornece as zonas e a região a serem usadas. O `netapp-san-us-east1` StorageClass cria PVCs no `san-backend-us-east1` back-end definido acima.

Passo 3: Criar e usar um PVC

Com o StorageClass criado e mapeado para um back-end, agora você pode criar PVCs.

Veja o exemplo `spec` abaixo:

```
---
kind: PersistentVolumeClaim
apiVersion: v1
metadata: null
name: pvc-san
spec: null
accessModes:
  - ReadWriteOnce
resources:
  requests:
    storage: 300Mi
storageClassName: netapp-san-us-east1
```

Criar um PVC usando este manifesto resultaria no seguinte:

```
kubectl create -f pvc.yaml
persistentvolumeclaim/pvc-san created
kubectl get pvc
NAME      STATUS    VOLUME   CAPACITY   ACCESS MODES   STORAGECLASS
AGE
pvc-san   Pending                                      netapp-san-us-east1
2s
```

```
kubectl describe pvc
Name:          pvc-san
Namespace:     default
StorageClass:  netapp-san-us-east1
Status:        Pending
Volume:
Labels:        <none>
Annotations:   <none>
Finalizers:    [kubernetes.io/pvc-protection]
Capacity:
Access Modes:
VolumeMode:    Filesystem
Mounted By:    <none>
Events:
  Type    Reason                Age   From                         Message
  ----    ------                ----  ----                         -------
  Normal  WaitForFirstConsumer  6s    persistentvolume-controller  waiting
for first consumer to be created before binding
```

Para o Trident criar um volume e vinculá-lo ao PVC, use o PVC em um pod. Veja o exemplo a seguir:

```
apiVersion: v1
kind: Pod
metadata:
  name: app-pod-1
spec:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: topology.kubernetes.io/region
            operator: In
            values:
            - us-east1
      preferredDuringSchedulingIgnoredDuringExecution:
      - weight: 1
        preference:
          matchExpressions:
          - key: topology.kubernetes.io/zone
            operator: In
            values:
            - us-east1-a
            - us-east1-b
  securityContext:
```

```
    runAsUser: 1000
    runAsGroup: 3000
    fsGroup: 2000
  volumes:
  - name: vol1
    persistentVolumeClaim:
      claimName: pvc-san
  containers:
  - name: sec-ctx-demo
    image: busybox
    command: [ "sh", "-c", "sleep 1h" ]
    volumeMounts:
    - name: vol1
      mountPath: /data/demo
    securityContext:
      allowPrivilegeEscalation: false
```

Este podSpec instrui o Kubernetes a agendar o pod em nós presentes na `us-east1` região e escolher entre qualquer nó presente nas `us-east1-a` zonas ou `us-east1-b`.

Veja a seguinte saída:

```
kubectl get pods -o wide
NAME        READY   STATUS    RESTARTS   AGE   IP               NODE
NOMINATED NODE   READINESS GATES
app-pod-1   1/1     Running   0          19s   192.168.25.131   node2
<none>           <none>
kubectl get pvc -o wide
NAME      STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS          AGE   VOLUMEMODE
pvc-san   Bound    pvc-ecb1e1a0-840c-463b-8b65-b3d033e2e62b   300Mi
RWO            netapp-san-us-east1   48s   Filesystem
```

Atualize os backends para incluir `supportedTopologies`

Os backends pré-existentes podem ser atualizados para incluir uma lista `supportedTopologies` de uso `tridentctl backend update`do . Isso não afetará os volumes que já foram provisionados e só será usado para PVCs subsequentes.

Encontre mais informações

- "Gerenciar recursos para contêineres"
- "NodeSeletor"
- "Afinidade e anti-afinidade"
- "Taints e Tolerations"

Trabalhe com snapshots do Trident

Os snapshots de volume do Kubernetes de volumes persistentes (PVS) permitem cópias pontuais de volumes. Você pode criar um snapshot de um volume criado usando o Trident, importar um snapshot criado fora do Trident, criar um novo volume a partir de um snapshot existente e recuperar dados de volume de snapshots.

Visão geral

O snapshot de volume é suportado por `ontap-nas` , `ontap-nas-flexgroup` , `ontap-san` , `ontap-san-economy` , `solidfire-san` , `azure-netapp-files` , e `google-cloud-netapp-volumes` motoristas.

Antes de começar

Você deve ter um controlador de snapshot externo e definições personalizadas de recursos (CRDs) para trabalhar com snapshots. Essa é a responsabilidade do orquestrador do Kubernetes (por exemplo: Kubeadm, GKE, OpenShift).

Se a distribuição do Kubernetes não incluir a controladora de snapshot e CRDs, Implantar um controlador de snapshot de volumeconsulte .

OBSERVAÇÃO	Não crie um controlador de snapshot se estiver criando instantâneos de volume sob demanda em um ambiente GKE. O GKE usa um controlador instantâneo oculto integrado.

Criar um instantâneo de volume

Passos

1. Criar um `VolumeSnapshotClass`. para obter mais informações, "VolumeSnapshotClass"consulte .
   - Os `driver` pontos para o driver Trident CSI.
   - `deletionPolicy` pode ser `Delete` ou `Retain`. Quando definido como `Retain`, o instantâneo físico subjacente no cluster de armazenamento é retido mesmo quando o `VolumeSnapshot` objeto é excluído.

   Exemplo

   ```
   cat snap-sc.yaml
   ```

   ```
   apiVersion: snapshot.storage.k8s.io/v1
   kind: VolumeSnapshotClass
   metadata:
     name: csi-snapclass
   driver: csi.trident.netapp.io
   deletionPolicy: Delete
   ```

2. Crie um instantâneo de um PVC existente.

   Exemplos

- Este exemplo cria um instantâneo de um PVC existente.

```
cat snap.yaml
```

```
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshot
metadata:
  name: pvc1-snap
spec:
  volumeSnapshotClassName: csi-snapclass
  source:
    persistentVolumeClaimName: pvc1
```

- Este exemplo cria um objeto instantâneo de volume para um PVC chamado `pvc1` e o nome do instantâneo é definido como `pvc1-snap`. Um VolumeSnapshot é análogo a um PVC e está associado a um `VolumeSnapshotContent` objeto que representa o snapshot real.

```
kubectl create -f snap.yaml
volumesnapshot.snapshot.storage.k8s.io/pvc1-snap created

kubectl get volumesnapshots
NAME                   AGE
pvc1-snap              50s
```

- Pode identificar o `VolumeSnapshotContent` objeto para o `pvc1-snap` VolumeSnapshot descrevendo-o. O `Snapshot Content Name` identifica o objeto VolumeSnapshotContent que serve este instantâneo. O `Ready To Use` parâmetro indica que o instantâneo pode ser usado para criar um novo PVC.

```
kubectl describe volumesnapshots pvc1-snap
Name:         pvc1-snap
Namespace:    default
...
Spec:
  Snapshot Class Name:    pvc1-snap
  Snapshot Content Name:  snapcontent-e8d8a0ca-9826-11e9-9807-
525400f3f660
  Source:
    API Group:
    Kind:       PersistentVolumeClaim
    Name:       pvc1
Status:
  Creation Time:  2019-06-26T15:27:29Z
```

```
  Ready To Use:   true
  Restore Size:   3Gi
...
```

Crie um PVC a partir de um instantâneo de volume

Você pode usar `dataSource` para criar um PVC usando um VolumeSnapshot nomeado `<pvc-name>` como a fonte dos dados. Depois que o PVC é criado, ele pode ser anexado a um pod e usado como qualquer outro PVC.

AVISO	O PVC será criado no mesmo backend que o volume de origem. "KB: A criação de um PVC a partir de um instantâneo de PVC do Trident não pode ser criada em um back-end alternativo"Consulte a .

O exemplo a seguir cria o PVC usando `pvc1-snap` como fonte de dados.

```
cat pvc-from-snap.yaml
```

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-from-snap
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: golden
  resources:
    requests:
      storage: 3Gi
  dataSource:
    name: pvc1-snap
    kind: VolumeSnapshot
    apiGroup: snapshot.storage.k8s.io
```

Importar um instantâneo de volume

O Trident oferece suporte ao "Processo de snapshot pré-provisionado do Kubernetes" para permitir que o administrador de cluster crie um `VolumeSnapshotContent` objeto e importe snapshots criados fora do Trident.

Antes de começar

O Trident deve ter criado ou importado o volume pai do instantâneo.

Passos

1. **Cluster admin:** Crie um `VolumeSnapshotContent` objeto que faça referência ao snapshot de back-end. Isso inicia o fluxo de trabalho de snapshot no Trident.

- Especifique o nome do instantâneo de back-end em `annotations` as `trident.netapp.io/internalSnapshotName: <"backend-snapshot-name">`.
- Especifique `<name-of-parent-volume-in-trident>/<volume-snapshot-content-name>` em `snapshotHandle`. esta é a única informação fornecida ao Trident pelo snapshotter externo na `ListSnapshots` chamada.

OBSERVAÇÃO	O `<volumeSnapshotContentName>` nem sempre pode corresponder ao nome do instantâneo do back-end devido a restrições de nomenclatura CR.

Exemplo

O exemplo a seguir cria um `VolumeSnapshotContent` objeto que faz referência a snapshot de back-end `snap-01` .

```
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshotContent
metadata:
  name: import-snap-content
  annotations:
    trident.netapp.io/internalSnapshotName: "snap-01"  # This is the
name of the snapshot on the backend
spec:
  deletionPolicy: Retain
  driver: csi.trident.netapp.io
  source:
    snapshotHandle: pvc-f71223b5-23b9-4235-bbfe-e269ac7b84b0/import-
snap-content # <import PV name or source PV name>/<volume-snapshot-
content-name>
  volumeSnapshotRef:
    name: import-snap
    namespace: default
```

2. **Cluster admin:** Crie o `VolumeSnapshot` CR que faz referência ao `VolumeSnapshotContent` objeto. Isso solicita acesso para usar o `VolumeSnapshot` em um namespace dado.

Exemplo

O exemplo a seguir cria um `VolumeSnapshot` CR chamado `import-snap` que faz referência ao `VolumeSnapshotContent` `import-snap-content` chamado .

```
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshot
metadata:
  name: import-snap
spec:
  # volumeSnapshotClassName: csi-snapclass (not required for pre-
provisioned or imported snapshots)
```

```
  source:
    volumeSnapshotContentName: import-snap-content
```

3. * Processamento interno (nenhuma ação necessária):* o Snapshotter externo reconhece o recém-criado `VolumeSnapshotContent` e executa a `ListSnapshots` chamada. Trident cria o `TridentSnapshot`.
   - O snapshotter externo define `VolumeSnapshotContent` para `readyToUse` e `VolumeSnapshot` para `true`.
   - Trident retorna `readyToUse=true`.
4. **Qualquer usuário:** Crie um `PersistentVolumeClaim` para fazer referência ao novo `VolumeSnapshot`, onde o `spec.dataSource` nome (ou `spec.dataSourceRef`) é o `VolumeSnapshot` nome.

   Exemplo

   O exemplo a seguir cria um PVC referenciando o `VolumeSnapshot` nome `import-snap`.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-from-snap
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: simple-sc
  resources:
    requests:
      storage: 1Gi
  dataSource:
    name: import-snap
    kind: VolumeSnapshot
    apiGroup: snapshot.storage.k8s.io
```

Recuperar dados de volume usando snapshots

O diretório instantâneo é oculto por padrão para facilitar a compatibilidade máxima dos volumes provisionados usando os `ontap-nas` drivers e `ontap-nas-economy`. Ative o `.snapshot` diretório para recuperar dados de instantâneos diretamente.

Use a CLI do ONTAP de restauração de snapshot de volume para restaurar um volume para um estado gravado em um snapshot anterior.

```
cluster1::*> volume snapshot restore -vserver vs0 -volume vol3 -snapshot
vol3_snap_archive
```

OBSERVAÇÃO Quando você restaura uma cópia snapshot, a configuração de volume existente é

sobrescrita. As alterações feitas aos dados de volume após a criação da cópia instantânea são perdidas.

Restauração de volume no local a partir de um instantâneo

O Trident fornece restauração rápida de volume no local a partir de um instantâneo usando o `TridentActionSnapshotRestore` CR (TASR). Esse CR funciona como uma ação imperativa do Kubernetes e não persiste após a conclusão da operação.

O Trident suporta a restauração de snapshots no `ontap-san` , `ontap-san-economy` , `ontap-nas` , `ontap-nas-flexgroup` , `azure-netapp-files` , `google-cloud-netapp-volumes` , e `solidfire-san` motoristas.

Antes de começar

Você deve ter um PVC vinculado e instantâneo de volume disponível.

- Verifique se o status do PVC está vinculado.

```
kubectl get pvc
```

- Verifique se o instantâneo do volume está pronto para ser usado.

```
kubectl get vs
```

Passos

1. Crie o TASR CR. Este exemplo cria um CR para instantâneo de PVC `pvc1` e volume `pvc1-snapshot` .

OBSERVAÇÃO O TAR CR deve estar num espaço de nomes onde o PVC e VS existam.

```
cat tasr-pvc1-snapshot.yaml
```

```
apiVersion: trident.netapp.io/v1
kind: TridentActionSnapshotRestore
metadata:
  name: trident-snap
  namespace: trident
spec:
  pvcName: pvc1
  volumeSnapshotName: pvc1-snapshot
```

2. Aplique o CR para restaurar a partir do instantâneo. Este exemplo restaura do instantâneo `pvc1`.

```
kubectl create -f tasr-pvc1-snapshot.yaml
```

```
tridentactionsnapshotrestore.trident.netapp.io/trident-snap created
```

Resultados

O Trident restaura os dados do snapshot. Você pode verificar o status de restauração de snapshot:

```
kubectl get tasr -o yaml
```

```
apiVersion: trident.netapp.io/v1
items:
- apiVersion: trident.netapp.io/v1
  kind: TridentActionSnapshotRestore
  metadata:
    creationTimestamp: "2023-04-14T00:20:33Z"
    generation: 3
    name: trident-snap
    namespace: trident
    resourceVersion: "3453847"
    uid: <uid>
  spec:
    pvcName: pvc1
    volumeSnapshotName: pvc1-snapshot
  status:
    startTime: "2023-04-14T00:20:34Z"
    completionTime: "2023-04-14T00:20:37Z"
    state: Succeeded
kind: List
metadata:
  resourceVersion: ""
```

OBSERVAÇÃO

- Na maioria dos casos, o Trident não tentará automaticamente a operação em caso de falha. Terá de efetuar novamente a operação.
- Os usuários do Kubernetes sem acesso de administrador podem ter permissão para que o administrador crie um TASR CR em seu namespace de aplicativo.

Eliminar um PV com instantâneos associados

Ao excluir um volume persistente com instantâneos associados, o volume Trident correspondente é atualizado para um "estado de exclusão". Remova os instantâneos de volume para excluir o volume Trident.

Implantar um controlador de snapshot de volume

Se a sua distribuição do Kubernetes não incluir a controladora de snapshot e CRDs, você poderá implantá-los da seguinte forma.

Passos

1. Criar CRDs de instantâneos de volume.

```
cat snapshot-setup.sh
```

```
#!/bin/bash
# Create volume snapshot CRDs
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
6.1/client/config/crd/snapshot.storage.k8s.io_volumesnapshotclasses.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
6.1/client/config/crd/snapshot.storage.k8s.io_volumesnapshotcontents.yam
l
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
6.1/client/config/crd/snapshot.storage.k8s.io_volumesnapshots.yaml
```

2. Crie o controlador instantâneo.

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-6.1/deploy/kubernetes/snapshot-
controller/rbac-snapshot-controller.yaml
```

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-6.1/deploy/kubernetes/snapshot-
controller/setup-snapshot-controller.yaml
```

OBSERVAÇÃO	Se necessário, abra `deploy/kubernetes/snapshot-controller/rbac-snapshot-controller.yaml` e atualize `namespace` para o seu namespace.

Links relacionados

- "Instantâneos de volume"
- "VolumeSnapshotClass"

Trabalhe com snapshots de grupo de volume do Trident

Snapshots de grupos de volumes do Kubernetes de Volumes Persistentes (PVs) O NetApp Trident oferece a capacidade de criar snapshots de vários volumes (um grupo de snapshots de volumes). Este snapshot de grupo de volumes representa cópias de vários volumes feitas no mesmo momento.

OBSERVAÇÃO	VolumeGroupSnapshot é um recurso beta do Kubernetes com APIs beta. O Kubernetes 1.32 é a versão mínima necessária para o VolumeGroupSnapshot.

Criar instantâneos de grupos de volumes

O recurso de snapshot de grupo de volumes é compatível com os seguintes drivers de armazenamento:

- `ontap-san`Driver - somente para os protocolos iSCSI e FC, não para o protocolo NVMe/TCP.
- `ontap-san-economy` - somente para o protocolo iSCSI.
- `ontap-nas`

OBSERVAÇÃO	O recurso de snapshot de grupo de volumes não é compatível com os sistemas de armazenamento NetApp ASA r2 ou AFX.

Antes de começar

- Certifique-se de que sua versão do Kubernetes seja K8s 1.32 ou superior.
- Você deve ter um controlador de snapshot externo e definições personalizadas de recursos (CRDs) para trabalhar com snapshots. Essa é a responsabilidade do orquestrador do Kubernetes (por exemplo: Kubeadm, GKE, OpenShift).

  Se a sua distribuição do Kubernetes não incluir o controlador de snapshot externo e os CRDs, consulte Implantar um controlador de snapshot de volume .

OBSERVAÇÃO	Não crie um controlador de snapshot se estiver criando snapshots de grupo de volumes sob demanda em um ambiente GKE. O GKE usa um controlador instantâneo oculto integrado.

- No controlador de snapshot YAML, defina o `CSIVolumeGroupSnapshot` defina o feature gate como 'true' para garantir que o instantâneo do grupo de volumes esteja habilitado.
- Crie as classes de instantâneos do grupo de volumes necessárias antes de criar um instantâneo do grupo de volumes.
- Certifique-se de que todos os PVCs/volumes estejam no mesmo SVM para poder criar o VolumeGroupSnapshot.

Passos

- Crie uma VolumeGroupSnapshotClass antes de criar um VolumeGroupSnapshot. Para obter mais informações, "Classe de instantâneo de grupo de volume"consulte .

```
apiVersion: groupsnapshot.storage.k8s.io/v1beta1
kind: VolumeGroupSnapshotClass
metadata:
  name: csi-group-snap-class
  annotations:
    kubernetes.io/description: "Trident group snapshot class"
driver: csi.trident.netapp.io
deletionPolicy: Delete
```

- Crie PVCs com rótulos necessários usando classes de armazenamento existentes ou adicione esses rótulos aos PVCs existentes.

  O exemplo a seguir cria o PVC usando `pvc1-group-snap` como fonte de dados e rótulo `consistentGroupSnapshot: groupA` . Defina a chave e o valor do rótulo com base em suas necessidades.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc1-group-snap
  labels:
    consistentGroupSnapshot: groupA
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 100Mi
  storageClassName: sc1-1
```

- Crie um VolumeGroupSnapshot com o mesmo rótulo (`consistentGroupSnapshot: groupA`) especificado no PVC.

  Este exemplo cria um instantâneo de grupo de volumes:

```
apiVersion: groupsnapshot.storage.k8s.io/v1beta1
kind: VolumeGroupSnapshot
metadata:
  name: "vgs1"
  namespace: trident
spec:
  volumeGroupSnapshotClassName: csi-group-snap-class
  source:
    selector:
      matchLabels:
        consistentGroupSnapshot: groupA
```

Recuperar dados de volume usando um snapshot de grupo

Você pode restaurar Volumes Persistentes individuais usando os snapshots individuais criados como parte do Snapshot do Grupo de Volumes. Não é possível recuperar o Snapshot do Grupo de Volumes como uma unidade.

Use a CLI do ONTAP de restauração de snapshot de volume para restaurar um volume para um estado gravado em um snapshot anterior.

```
cluster1::*> volume snapshot restore -vserver vs0 -volume vol3 -snapshot
vol3_snap_archive
```

OBSERVAÇÃO	Quando você restaura uma cópia snapshot, a configuração de volume existente é sobrescrita. As alterações feitas aos dados de volume após a criação da cópia instantânea são perdidas.

Restauração de volume no local a partir de um instantâneo

O Trident fornece restauração rápida de volume no local a partir de um instantâneo usando o `TridentActionSnapshotRestore` CR (TASR). Esse CR funciona como uma ação imperativa do Kubernetes e não persiste após a conclusão da operação.

Para obter mais informações, "Restauração de volume no local a partir de um instantâneo"consulte .

Excluir um PV com snapshots de grupo associados

Ao excluir um instantâneo de volume de grupo:

- Você pode excluir VolumeGroupSnapshots como um todo, não snapshots individuais no grupo.
- Se PersistentVolumes forem excluídos enquanto houver um snapshot para esse PersistentVolume, o Trident moverá esse volume para um estado de "exclusão" porque o snapshot deve ser removido antes que o volume possa ser removido com segurança.
- Se um clone tiver sido criado usando um instantâneo agrupado e o grupo precisar ser excluído, uma operação de divisão no clone será iniciada e o grupo não poderá ser excluído até que a divisão seja concluída.

Implantar um controlador de snapshot de volume

Se a sua distribuição do Kubernetes não incluir a controladora de snapshot e CRDs, você poderá implantá-los da seguinte forma.

Passos

1. Criar CRDs de instantâneos de volume.

```
cat snapshot-setup.sh
```

```
#!/bin/bash
# Create volume snapshot CRDs
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
8.2/client/config/crd/groupsnapshot.storage.k8s.io_volumegroupsnapshotcl
asses.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
8.2/client/config/crd/groupsnapshot.storage.k8s.io_volumegroupsnapshotco
```

```
ntents.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
8.2/client/config/crd/groupsnapshot.storage.k8s.io_volumegroupsnapshots.
yaml
```

2. Crie o controlador instantâneo.

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-8.2/deploy/kubernetes/snapshot-
controller/rbac-snapshot-controller.yaml
```

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-8.2/deploy/kubernetes/snapshot-
controller/setup-snapshot-controller.yaml
```

OBSERVAÇÃO	Se necessário, abra `deploy/kubernetes/snapshot-controller/rbac-snapshot-controller.yaml` e atualize `namespace` para o seu namespace.

Links relacionados

- "Classe de instantâneo de grupo de volume"
- "Instantâneos de volume"

Gerenciar e monitorar o Trident

Atualize o Trident

Atualize o Trident

Começando com o lançamento de 24,02, o Trident segue uma cadência de lançamento de quatro meses, oferecendo três grandes lançamentos a cada ano civil. Cada nova versão baseia-se nas versões anteriores e fornece novos recursos, melhorias de desempenho, correções de bugs e melhorias. Recomendamos que você atualize pelo menos uma vez por ano para aproveitar os novos recursos do Trident.

Considerações antes da atualização

Ao atualizar para a versão mais recente do Trident, considere o seguinte:

- Deve haver apenas uma instância do Trident instalada em todos os namespaces em um determinado cluster do Kubernetes.
- O Trident 23,07 e posterior requer instantâneos de volume v1 e não suporta mais instantâneos alfa ou beta.
- Ao atualizar, é importante que você forneça `parameter.fsType` em `StorageClasses` usado pelo Trident. Você pode excluir e recriar `StorageClasses` sem interromper volumes pré-existentes.
  - Este é um **requisito** para aplicação de "contextos de segurança" volumes SAN.
  - O diretório https://github.com/NetApp https://github.com/NetApp/Trident Trident/tree/master/Trident Trident-installer/sample-input[sample input] contém exemplos, como https://github.com/NetApp/Trident/blob/master/Trident-installer/sample-input/storage-class-samples/storage-class-[`storage-class-bronze-default.yaml`basic[`storage-class-basic.yaml.templ`
  - Para obter mais informações, "Problemas conhecidos"consulte .

Passo 1: Selecione uma versão

As versões do Trident seguem uma convenção de nomenclatura baseada em data `YY.MM`, onde "YY" é os últimos dois dígitos do ano e "MM" é o mês. Os lançamentos de ponto seguem uma `YY.MM.X` convenção, onde "X" é o nível de patch. Você selecionará a versão para a qual atualizar com base na versão da qual você está atualizando.

- Você pode fazer uma atualização direta para qualquer versão de destino que esteja dentro de uma janela de quatro versões da versão instalada. Por exemplo, você pode atualizar diretamente de 24,06 (ou qualquer lançamento de 24,06 pontos) para 25,06.
- Se você estiver atualizando de uma versão fora da janela de quatro versões, execute uma atualização em várias etapas. Use as instruções de atualização do "versão anterior"para atualizar para a versão mais recente que se encaixa na janela de quatro versões. Por exemplo, se você estiver executando o 23,07 e quiser atualizar para o 25,06:
  a. Primeiro upgrade de 23,07 para 24,06.
  b. Em seguida, atualize de 24,06 para 25,06.

Passo 2: Determine o método de instalação original

Para determinar qual versão você usou para instalar o Trident originalmente:

1. Use `kubectl get pods -n trident` para examinar os pods.
   - Se não houver nenhum pod do operador, o Trident foi instalado usando `tridentctl`o .
   - Se houver um pod do operador, o Trident foi instalado usando o operador Trident manualmente ou usando o Helm.
2. Se houver um pod do operador, use `kubectl describe torc` para determinar se o Trident foi instalado usando o Helm.
   - Se houver uma etiqueta Helm, o Trident foi instalado usando Helm.
   - Se não houver nenhuma etiqueta Helm, o Trident foi instalado manualmente usando o operador Trident.

Passo 3: Selecione um método de atualização

Geralmente, você deve atualizar usando o mesmo método que usou para a instalação inicial, porém você pode "mova entre os métodos de instalação". Existem duas opções para atualizar Trident.

- "Atualize usando o operador Trident"

> **DICA** Sugerimos que você revise "Compreender o fluxo de trabalho de atualização do operador" antes de atualizar com o operador.

*

Atualize com o operador

Entenda o fluxo de trabalho de atualização do operador Trident

Antes de usar o operador Trident para atualizar o Trident, você deve entender os processos em segundo plano que ocorrem durante a atualização. Isso inclui alterações no controlador Trident, no pod de nó e no pod de nó e no DaemonSet que permitem atualizações contínuas.

Manuseio de atualização do operador Trident

Uma das muitas "Benefícios de usar o operador Trident" maneiras de instalar e atualizar Trident é o gerenciamento automático de objetos do Trident e do Kubernetes sem interromper os volumes montados existentes. Dessa forma, Trident pode suportar atualizações sem inatividade, ou "*atualizações contínuas*". Em particular, o operador do Trident se comunica com o cluster Kubernetes para:

- Exclua e recrie a implantação do controlador Trident e o nó DaemonSet.
- Substitua o pod de nó Trident e o pod de nó Trident por novas versões.
  - Se um nó não for atualizado, ele não impedirá que os nós restantes sejam atualizados.
  - Somente os nós com um pod de nó Trident em execução podem montar volumes.

> **DICA** Para obter mais informações sobre a arquitetura do Trident no cluster do Kubernetes, "Arquitetura da Trident"consulte .

Fluxo de trabalho de atualização do operador

Quando você inicia uma atualização usando o operador Trident:

1. O operador **Trident**:
   a. Detecta a versão atualmente instalada do Trident (versão *n*).
   b. Atualiza todos os objetos Kubernetes, incluindo CRDs, RBAC e Trident SVC.
   c. Exclui a implantação do controlador Trident para a versão *n*.
   d. Cria a implantação do controlador Trident para a versão *n-1*.
2. O Kubernetes* cria o pod de controlador Trident para *n-1*.
3. O operador **Trident**:
   a. Exclui o nó Trident DaemonSet para *n*. O operador não espera o encerramento do Node Pod.
   b. Cria o nó Trident Daemonset para *n-1*.
4. **Kubernetes** cria pods de nós do Trident em nós que não executam o Pod de nó do Trident *n*. Isso garante que nunca mais de um pod de nó Trident, de qualquer versão, em um nó.

Atualize uma instalação do Trident usando o operador Trident ou Helm

Você pode atualizar o Trident usando o operador Trident, seja manualmente ou usando o Helm. Você pode atualizar de uma instalação do operador Trident para outra instalação do operador Trident ou atualizar de uma `tridentctl` instalação para uma versão do operador Trident. Revise "Selecione um método de atualização" antes de atualizar uma instalação do operador Trident.

Atualize uma instalação manual

Você pode atualizar de uma instalação do operador Trident com escopo de cluster para outra instalação do operador Trident com escopo de cluster. Todas as versões do Trident usam um operador com escopo de cluster.

OBSERVAÇÃO	Para atualizar do Trident que foi instalado usando o operador com escopo de namespace (versões 20,07 a 20,10), use as instruções de atualização do "sua versão instalada"Trident.

Sobre esta tarefa

O Trident fornece um arquivo de pacote que você pode usar para instalar o operador e criar objetos associados para sua versão do Kubernetes.

- Para clusters que executam o Kubernetes 1,25 ou posterior, "bundle_post_1_25.yaml"use o .

Antes de começar

Verifique se você está usando um cluster do Kubernetes executando "Uma versão compatível do Kubernetes"o .

Passos

1. Verifique sua versão do Trident:

```
./tridentctl -n trident version
```

2. Atualizar o `operator.yaml` , `tridentorchestrator_cr.yaml` , e `post_1_25_bundle.yaml` com o registro e os caminhos de imagem para a versão para a qual você está atualizando (por exemplo, 25.06) e o segredo correto.
3. Exclua o operador Trident que foi usado para instalar a instância atual do Trident . Por exemplo, se você estiver atualizando da versão 25.02, execute o seguinte comando:

```
kubectl delete -f 25.02.0/trident-installer/deploy/<bundle.yaml> -n
trident
```

4. Se você personalizou sua instalação inicial usando `TridentOrchestrator` atributos, você pode editar o `TridentOrchestrator` objeto para modificar os parâmetros de instalação. Isso pode incluir alterações feitas para especificar Registros de imagens Trident e CSI espelhados para o modo offline, habilitar logs de depuração ou especificar segredos de recebimento de imagens.
5. Instale o Trident usando o arquivo YAML do pacote correto para seu ambiente, onde *<bundle.yaml>* é `bundle_pre_1_25.yaml` ou `bundle_post_1_25.yaml` com base na sua versão do Kubernetes. Por exemplo, se você estiver instalando o Trident 25.06.0, execute o seguinte comando:

```
kubectl create -f 25.06.0/trident-installer/deploy/<bundle.yaml> -n
trident
```

6. Edite o torque do tridente para incluir a imagem 25.06.0.

Atualize uma instalação do Helm

Você pode atualizar uma instalação do Trident Helm.

AVISO Ao atualizar um cluster do Kubernetes do 1,24 para o 1,25 ou posterior que tenha o Trident instalado, você deve atualizar o Values.yaml para definir `excludePodSecurityPolicy true` ou adicionar `--set excludePodSecurityPolicy=true helm upgrade` ao comando antes de atualizar o cluster.

Se você já atualizou seu cluster do Kubernetes de 1,24 para 1,25 sem atualizar o leme do Trident, a atualização do leme falhará. Para que a atualização do leme passe, execute estes passos como pré-requisitos:

1. Instale o plugin Helm-mapkubeapis https://github.com/helm/helm-mapkubeapis do .
2. Execute uma execução a seco para a versão Trident no namespace onde o Trident está instalado. Isso lista os recursos, que serão limpos.

```
helm mapkubeapis --dry-run trident  --namespace trident
```

3. Execute uma corrida completa com o leme para fazer a limpeza.

```
helm mapkubeapis trident  --namespace trident
```

Passos

1. Se você "Trident instalado usando Helm", pode usar `helm upgrade trident netapp-trident/trident-operator --version 100.2610.0` para atualizar em uma única etapa. Se você não adicionou o repositório Helm ou não consegue usá-lo para atualizar:
   a. Transfira a versão mais recente do Trident a partir de "A seção *assets* no GitHub".
   b. Utilize o `helm upgrade` comando onde `trident-operator-26.10.1.tgz` reflete a versão para a qual você deseja atualizar.

```
helm upgrade <name> trident-operator-26.10.1.tgz
```

OBSERVAÇÃO	Se você definir opções personalizadas durante a instalação inicial (como especificar Registros privados e espelhados para imagens Trident e CSI), anexe o `helm upgrade` comando usando `--set` para garantir que essas opções estejam incluídas no comando upgrade, caso contrário, os valores serão redefinidos para padrão.

2. Execute `helm list` para verificar se o gráfico e a versão do aplicativo foram atualizados. Execute `tridentctl logs` para rever todas as mensagens de depuração.

Atualize de uma `tridentctl` instalação para o operador Trident

Pode atualizar para a versão mais recente do operador Trident a partir de uma `tridentctl` instalação. Os backends e PVCs existentes estarão automaticamente disponíveis.

OBSERVAÇÃO	Antes de alternar entre os métodos de instalação, revise "Movendo-se entre os métodos de instalação".

Passos

1. Transfira a versão mais recente do Trident.

```
# Download the release required [26.10.1]
mkdir 26.10.1
cd 26.10.1
wget
https://github.com/NetApp/trident/releases/download/v26.10.0/trident-
installer-26.10.1.tar.gz
tar -xf trident-installer-26.10.1.tar.gz
cd trident-installer
```

2. Crie o `tridentorchestrator` CRD a partir do manifesto.

```
kubectl create -f
```

```
deploy/crds/trident.netapp.io_tridentorchestrators_crd_post1.16.yaml
```

3. Implante o operador com escopo de cluster no mesmo namespace.

```
kubectl create -f deploy/<bundle-name.yaml>

serviceaccount/trident-operator created
clusterrole.rbac.authorization.k8s.io/trident-operator created
clusterrolebinding.rbac.authorization.k8s.io/trident-operator created
deployment.apps/trident-operator created
podsecuritypolicy.policy/tridentoperatorpods created

#Examine the pods in the Trident namespace
NAME                                  READY   STATUS    RESTARTS   AGE
trident-controller-79df798bdc-m79dc   6/6     Running   0          150d
trident-node-linux-xrst8              2/2     Running   0          150d
trident-operator-5574dbbc68-nthjv     1/1     Running   0          1m30s
```

4. Crie um `TridentOrchestrator` CR para instalar o Trident.

```
cat deploy/crds/tridentorchestrator_cr.yaml
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident

kubectl create -f deploy/crds/tridentorchestrator_cr.yaml

#Examine the pods in the Trident namespace
NAME                                READY   STATUS    RESTARTS   AGE
trident-csi-79df798bdc-m79dc        6/6     Running   0          1m
trident-csi-xrst8                   2/2     Running   0          1m
trident-operator-5574dbbc68-nthjv   1/1     Running   0          5m41s
```

5. Confirme se o Trident foi atualizado para a versão pretendida.

```
kubectl describe torc trident | grep Message -A 3

Message:                Trident installed
Namespace:              trident
```

```
Status:                 Installed
Version:                v26.10.0
```

Atualize o Trident com tridentctl

Você pode atualizar facilmente uma instalação existente do Trident usando `tridentctl`o .

Sobre esta tarefa

Desinstalar e reinstalar o Trident funciona como uma atualização. Quando você desinstalar o Trident, a reivindicação de volume persistente (PVC) e o volume persistente (PV) usados pela implantação do Trident não são excluídos. Os PVS que já foram provisionados permanecerão disponíveis enquanto o Trident estiver off-line, e o Trident provisionará volumes para quaisquer PVCs que forem criados no período intermediário depois que ele estiver novamente on-line.

Antes de começar

Analise "Selecione um método de atualização" antes de atualizar usando `tridentctl`.

Passos

1. Execute o comando uninstall `tridentctl` para remover todos os recursos associados ao Trident, exceto para os CRDs e objetos relacionados.

```
./tridentctl uninstall -n <namespace>
```

2. Reinstale Trident. Consulte "Instale o Trident usando o tridentctl".

IMPORTANTE	Não interrompa o processo de atualização. Certifique-se de que o instalador é executado até a conclusão.

Gerenciar o Trident usando o tridentctl

O "Pacote de instalação do Trident" inclui o `tridentctl` utilitário de linha de comando para fornecer acesso simples ao Trident. Os usuários do Kubernetes com Privileges suficientes podem usá-lo para instalar o Trident ou gerenciar o namespace que contém o pod Trident.

Comandos e sinalizadores globais

Você pode executar `tridentctl help` para obter uma lista de comandos disponíveis `tridentctl` ou anexar o `--help` sinalizador a qualquer comando para obter uma lista de opções e sinalizadores para esse comando específico.

```
tridentctl [command] [--optional-flag]
```

O utilitário Trident `tridentctl` suporta os seguintes comandos e sinalizadores globais.

▼ Comandos

create
Adicione um recurso ao Trident.

delete
Remova um ou mais recursos do Trident.

get
Obtenha um ou mais recursos do Trident.

help
Ajuda sobre qualquer comando.

images
Imprima uma tabela das imagens de contentor que o Trident necessita.

import
Importar um recurso existente para o Trident.

install
Instale o Trident.

logs
Imprimir os registos a partir do Trident.

send
Enviar um recurso do Trident.

uninstall
Desinstale o Trident.

update
Modificar um recurso no Trident.

update backend state
Suspender temporariamente as operações de back-end.

upgrade
Atualizar um recurso no Trident.

version
Imprima a versão do Trident.

▼ **Bandeiras globais**

-d, --debug
Saída de depuração.

-h, --help
Ajuda para `tridentctl`.

-k, --kubeconfig string

Especifique `KUBECONFIG` o caminho para executar comandos localmente ou de um cluster do Kubernetes para outro.

OBSERVAÇÃO	Como alternativa, você pode exportar a `KUBECONFIG` variável para apontar para um cluster Kubernetes específico e emitir `tridentctl` comandos para esse cluster.

-n, --namespace string

Namespace da implantação do Trident.

-o, --output string

Formato de saída. Um de JSON|yaml|name|wide|ps (padrão).

-s, --server string

Endereço/porta da interface REST do Trident.

AVISO	A interface REST DO Trident pode ser configurada para ouvir e servir apenas em 127.0.0.1 (para IPv4) ou [::1] (para IPv6).

Opções de comando e sinalizadores

criar

Use o `create` comando para adicionar um recurso ao Trident.

```
tridentctl create [option]
```

Opções

`backend`: Adicione um backend ao Trident.

eliminar

Use o `delete` comando para remover um ou mais recursos do Trident.

```
tridentctl delete [option]
```

Opções

`backend`: Excluir um ou mais backends de armazenamento do Trident.
`snapshot`: Excluir um ou mais snapshots de volume do Trident.
`storageclass`: Excluir uma ou mais classes de armazenamento do Trident.
`volume`: Excluir um ou mais volumes de armazenamento do Trident.

obter

Use o `get` comando para obter um ou mais recursos do Trident.

```
tridentctl get [option]
```

Opções

`backend`: Obtenha um ou mais backends de armazenamento do Trident.
`snapshot`: Obter um ou mais snapshots do Trident.
`storageclass`: Obtenha uma ou mais classes de armazenamento do Trident.
`volume`: Obtenha um ou mais volumes do Trident.

Bandeiras

`-h`, `--help`: Ajuda para volumes.
`--parentOfSubordinate string`: Limitar consulta ao volume de origem subordinado.
`--subordinateOf string`: Limitar consulta a subordinados de volume.

imagens

Use `images` sinalizadores para imprimir uma tabela das imagens de contentor que o Trident precisa.

```
tridentctl images [flags]
```

Bandeiras

`-h`, `--help`: Ajuda para imagens.
`-v --k8s-version string`, : Versão semântica do cluster do Kubernetes.

importar volume

Use o `import volume` comando para importar um volume existente para o Trident.

```
tridentctl import volume <backendName> <volumeName> [flags]
```

Aliases

`volume`, `v`

Bandeiras

`-f --filename string`, : Caminho para o arquivo PVC YAML ou JSON.
`-h`, `--help`: Ajuda para volume.
`--no-manage`: Criar apenas PV/PVC. Não assuma o gerenciamento do ciclo de vida do volume.

instale

Use os `install` sinalizadores para instalar o Trident.

```
tridentctl install [flags]
```

Bandeiras

`--autosupport-image string`: A imagem do contêiner para o Autosupport Telemetry (padrão "netapp/trident autosupport:<current-version>").
`--autosupport-proxy string` : O endereço/porta de um proxy para envio de Telemetria Autosupport.
`--enable-node-prep` : Tentar instalar os pacotes necessários nos nós.
`--generate-custom-yaml` : Gere arquivos YAML sem instalar nada.
`-h` , `--help` : Ajuda para instalação.
`--http-request-timeout` : Substituir o tempo limite de solicitação HTTP para a API REST do controlador Trident (padrão 1m30s).
`--image-registry string` : O endereço/porta de um registro de imagem interno.

`--k8s-timeout duration` : O tempo limite para todas as operações do Kubernetes (padrão 3m0s).
`--kubelet-dir string` : O local do host do estado interno do kubelet (padrão "/var/lib/kubelet").
`--log-format string` : O formato de registro do Trident (texto, json) (padrão "texto").
`--node-prep` : Permite que o Trident prepare os nós do cluster Kubernetes para gerenciar volumes usando o protocolo de armazenamento de dados especificado. **Atualmente, `iscsi` é o único valor suportado. A partir do OpenShift 4.19, a versão mínima do Trident suportada para esse recurso é 25.06.1.**
`--pv string` : O nome do PV legado usado pelo Trident garante que isso não exista (padrão "trident").
`--pvc string` : O nome do PVC legado usado pelo Trident garante que isso não exista (padrão "trident").
`--silence-autosupport` : Não enviar pacotes de suporte automático para a NetApp automaticamente (padrão verdadeiro).
`--silent` : Desabilita a maioria das saídas durante a instalação.
`--trident-image string` : A imagem do Trident para instalar.
`--k8s-api-qps` : O limite de consultas por segundo (QPS) para solicitações da API do Kubernetes (padrão 100; opcional).
`--use-custom-yaml` : Use qualquer arquivo YAML existente no diretório de configuração.
`--use-ipv6` : Use IPv6 para comunicação do Trident.

registos

Use `logs` sinalizadores para imprimir os logs do Trident.

```
tridentctl logs [flags]
```

Bandeiras

`-a`, `--archive`: Crie um arquivo de suporte com todos os logs, a menos que especificado de outra forma.
`-h --help`, : Ajuda para logs.
`-l --log string`, : Trident log a ser exibido. Um dos Trident|auto|Trident-operator|All (predefinição "auto").
`--node string`: O nome do nó Kubernetes do qual você pode coletar logs do pod de nó.
`-p --previous`, : Obtém os registos para a instância de contentor anterior, se existir.
`--sidecars`: Obter os logs para os recipientes sidecar.

enviar

Use o `send` comando para enviar um recurso do Trident.

```
tridentctl send [option]
```

Opções

`autosupport`: Enviar um arquivo AutoSupport para o NetApp.

desinstalar

Use `uninstall` sinalizadores para desinstalar o Trident.

```
tridentctl uninstall [flags]
```

Bandeiras

`-h, --help`: Ajuda para desinstalar.
`--silent`: Desativar a saída MOST durante a desinstalação.

atualização

Use o `update` comando para modificar um recurso no Trident.

```
tridentctl update [option]
```

Opções

`backend`: Atualize um backend no Trident.

atualizar estado de back-end

Use o `update backend state` comando para suspender ou retomar as operações de back-end.

```
tridentctl update backend state <backend-name> [flag]
```

Pontos a considerar

- Se um back-end for criado usando um TridentBackendConfig (tbc), o back-end não poderá ser atualizado usando um `backend.json` arquivo.
- Se o `userState` foi definido em um tbc, ele não pode ser modificado usando o `tridentctl update backend state <backend-name> --user-state suspended/normal` comando.
- Para recuperar a capacidade de definir a `userState` via tridentctl depois de ter sido definida via tbc, o `userState` campo deve ser removido do tbc. Isso pode ser feito usando o `kubectl edit tbc` comando. Depois que o `userState` campo for removido, você pode usar o `tridentctl update backend state` comando para alterar o `userState` de um backend.
- Utilize os `tridentctl update backend state` para alterar o `userState`. Você também pode atualizar o `userState` usando `TridentBackendConfig` ou `backend.json` arquivo; isso aciona uma reinicialização completa do back-end e pode ser demorado.

  #### Bandeiras

  `-h` `--help`, : Ajuda para o estado de back-end.
  `--user-state`: Defina como `suspended` para pausar operações de back-end. Defina como `normal` para retomar as operações de back-end. Quando definido para `suspended`:

- `AddVolume` e `Import Volume` estão em pausa.
- `CloneVolume` `ResizeVolume`, , `PublishVolume` `UnPublishVolume` , `CreateSnapshot`, , , `GetSnapshot` `RestoreSnapshot`, , `DeleteSnapshot` `RemoveVolume` , `GetVolumeExternal`, , , `ReconcileNodeAccess` permanecer disponível.

Você também pode atualizar o estado de back-end usando `userState` o campo no arquivo de configuração de back-end `TridentBackendConfig` ou `backend.json`. Para obter mais informações, "Opções para gerenciar backends"consulte e "Execute o gerenciamento de back-end com o kubectl".

Exemplo:

JSON

Siga estas etapas para atualizar o `userState` usando o `backend.json` arquivo:

1. Edite o `backend.json` arquivo para incluir o `userState` campo com o seu valor definido como 'uspended'.

2. Atualize o backend usando o `tridentctl update backend` comando e o caminho para o atualizado `backend.json` arquivo.

   Exemplo: `tridentctl update backend -f /<path to backend JSON file>/backend.json -n trident`

```
{
  "version": 1,
  "storageDriverName": "ontap-nas",
  "managementLIF": "<redacted>",
  "svm": "nas-svm",
  "backendName": "customBackend",
  "username": "<redacted>",
  "password": "<redacted>",
  "userState": "suspended"
}
```

YAML

Você pode editar o tbc depois que ele foi aplicado usando o `kubectl edit <tbc-name> -n <namespace>` comando. O exemplo a seguir atualiza o estado de back-end para suspender usando a `userState: suspended` opção:

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-ontap-nas
spec:
  version: 1
  backendName: customBackend
  storageDriverName: ontap-nas
  managementLIF: <redacted>
  svm: nas-svm
  userState: suspended
  credentials:
    name: backend-tbc-ontap-nas-secret
```

versão

Use `version` sinalizadores para imprimir a versão do `tridentctl` e o serviço Trident em execução.

```
tridentctl version [flags]
```

Bandeiras

`--client`: Somente versão do cliente (nenhum servidor necessário).
`-h, --help`: Ajuda para a versão.

Suporte ao plugin

O Trigentctl suporta plugins semelhantes ao kubectl. O tridentctl deteta um plugin se o nome do arquivo binário do plugin seguir o esquema "tridentctl-<plugin>", e o binário está localizado em uma pasta listada a variável de ambiente PATH. Todos os plugins detetados estão listados na seção plugin da ajuda do tridentctl. Opcionalmente, você também pode limitar a pesquisa especificando uma pasta de plug-in na variável de enviornment TRIDENCTL_PLUGIN_PATH (exemplo: `TRIDENTCTL_PLUGIN_PATH=~/tridentctl-plugins/`). Se a variável for usada, tridenctl pesquisará somente na pasta especificada.

Monitorar as operações de storage do Trident

O Trident fornece um conjunto de endpoints de métricas Prometheus que você pode usar para monitorar o desempenho do Trident.

Visão geral

As métricas fornecidas pelo Trident permitem que você faça o seguinte:

- Mantenha o controle sobre a integridade e a configuração do Trident. Você pode examinar como as operações são bem-sucedidas e se elas podem se comunicar com os backends como esperado.
- Examine as informações de uso do back-end e entenda quantos volumes são provisionados em um back-end e a quantidade de espaço consumido, etc.
- Mantenha um mapeamento da quantidade de volumes provisionados em backends disponíveis.
- Acompanhe o desempenho. Você pode dar uma olhada em quanto tempo leva para o Trident se comunicar com backends e realizar operações.

OBSERVAÇÃO	Por padrão, as métricas do Trident são expostas na porta de destino. `8001` no `/metrics` ponto final. Essas métricas são **ativadas por padrão** quando o Trident é instalado. Você pode configurar o consumo de métricas do Trident via HTTPS na porta. `8444` também.

O que você vai precisar

- Um cluster do Kubernetes com o Trident instalado.
- Uma instância Prometheus. Isso pode ser um "Implantação do Prometheus em contêiner" ou você pode optar por executar Prometheus como um "aplicação nativa".

Passo 1: Defina um alvo Prometheus

Você deve definir um alvo do Prometheus para coletar as métricas e obter informações sobre os backends gerenciados Trident , os volumes que ele cria e assim por diante. Ver"Documentação do operador Prometheus" .

Passo 2: Crie um Prometheus ServiceMonitor

Para consumir as métricas do Trident, você deve criar um Prometheus ServiceMonitor que vigia `trident-csi` o serviço e escuta na `metrics` porta. Um exemplo de ServiceMonitor se parece com isso:

```
apiVersion: monitoring.coreos.com/v1
```

```
kind: ServiceMonitor
metadata:
  name: trident-sm
  namespace: monitoring
  labels:
    release: prom-operator
spec:
  jobLabel: trident
  selector:
    matchLabels:
      app: controller.csi.trident.netapp.io
  namespaceSelector:
    matchNames:
      - trident
  endpoints:
    - port: metrics
      interval: 15s
```

Esta definição de ServiceMonitor recupera métricas retornadas pelo `trident-csi` serviço e procura especificamente por `metrics` ponto final do serviço. Como resultado, o Prometheus agora está configurado para entender as métricas do Trident.

Além das métricas disponíveis diretamente do Trident, o kubelet expõe muitas `kubelet_volume_*` métricas por meio do seu próprio ponto de extremidade de métricas. O Kubelet pode fornecer informações sobre os volumes anexados e pods e outras operações internas que ele manipula. Consulte a "aqui".

Consuma métricas do Trident via HTTPS.

Para consumir métricas do Trident via HTTPS (porta 8444), você deve modificar a definição do ServiceMonitor para incluir a configuração TLS. Você também precisa copiar o `trident-csi` segredo do `trident` namespace para o namespace onde o Prometheus está sendo executado. Você pode fazer isso usando o seguinte comando:

```
kubectl get secret trident-csi -n trident -o yaml | sed 's/namespace:
trident/namespace: monitoring/' | kubectl apply -f -
```

Um exemplo de ServiceMonitor para métricas HTTPS se parece com isto:

```
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  name: trident-sm
  namespace: monitoring
  labels:
    release: prom-operator
spec:
  jobLabel: trident
  selector:
```

```
    matchLabels:
      app: controller.csi.trident.netapp.io
  namespaceSelector:
    matchNames:
      - trident
  endpoints:
    - interval: 15s
      path: /metrics
      port: https-metrics
      scheme: https
      tlsConfig:
        ca:
          secret:
            key: caCert
            name: trident-csi
        cert:
          secret:
            key: clientCert
            name: trident-csi
        keySecret:
          key: clientKey
          name: trident-csi
        serverName: trident-csi
```

O Trident oferece suporte a métricas HTTPS em todos os métodos de instalação: tridentctl, Helm chart e Operator.

- Se você estiver usando o `tridentctl install` comando, você pode passar o `--https-metrics` sinalizador para ativar as métricas HTTPS.
- Se você estiver usando o gráfico Helm, poderá definir o `httpsMetrics` parâmetro para habilitar métricas HTTPS.
- Se você estiver usando arquivos YAML, poderá adicionar o `--https_metrics` bandeira para o `trident-main` recipiente no `trident-deployment.yaml` arquivo.

Passo 3: Consultar métricas do Trident com PromQL

PromQL é bom para criar expressões que retornam dados de séries temporais ou tabulares.

Aqui estão algumas consultas PromQL que você pode usar:

Obtenha informações de saúde do Trident

- **Porcentagem de respostas HTTP 2XX do Trident**

```
(sum (trident_rest_ops_seconds_total_count{status_code=~"2.."} OR on()
vector(0)) / sum (trident_rest_ops_seconds_total_count)) * 100
```

- **Porcentagem de respostas REST do Trident via código de status**

```
(sum (trident_rest_ops_seconds_total_count) by (status_code)  / scalar
(sum (trident_rest_ops_seconds_total_count))) * 100
```

- **Duração média em ms das operações realizadas pela Trident**

```
sum by (operation)
(trident_operation_duration_milliseconds_sum{success="true"}) / sum by
(operation)
(trident_operation_duration_milliseconds_count{success="true"})
```

Obtenha informações de uso do Trident

- **Tamanho médio do volume**

```
trident_volume_allocated_bytes/trident_volume_count
```

- **Espaço total de volume provisionado por cada back-end**

```
sum (trident_volume_allocated_bytes) by (backend_uuid)
```

Obtenha uso de volume individual

OBSERVAÇÃO	Isso é ativado somente se as métricas do kubelet também forem coletadas.

- **Porcentagem de espaço usado para cada volume**

```
kubelet_volume_stats_used_bytes / kubelet_volume_stats_capacity_bytes *
100
```

Saiba mais sobre telemetria Trident AutoSupport

Por padrão, o Trident envia métricas de Prometheus e informações básicas de back-end para o NetApp em uma cadência diária.

- Para impedir que o Trident envie métricas do Prometheus e informações básicas de back-end para o NetApp, passe a `--silence-autosupport` bandeira durante a instalação do Trident.
- O Trident também pode enviar Registros de contentores para o suporte da NetApp sob demanda por meio `tridentctl send autosupport`do . Você precisará acionar o Trident para fazer o upload dos seus logs. Antes de enviar logs, você deve aceitar o NetApp'"política de privacidade"s .
- A menos que especificado, o Trident obtém os logs das últimas 24 horas.

- Você pode especificar o período de tempo de retenção do log com o `--since` sinalizador. Por exemplo `tridentctl send autosupport --since=1h`: . Essas informações são coletadas e enviadas por meio de um `trident-autosupport` contentor que é instalado ao lado do Trident. Pode obter a imagem do contentor em "Trident AutoSupport".
- Trident AutoSupport não coleta nem transmite Informações de Identificação Pessoal (PII) ou Informações Pessoais. Vem com uma "EULA" que não se aplica à própria imagem do contêiner Trident. Você pode saber mais sobre o compromisso da NetApp com a segurança e a confiança dos dados "aqui".

Um exemplo de payload enviado pela Trident é assim:

```
---
items:
  - backendUUID: ff3852e1-18a5-4df4-b2d3-f59f829627ed
    protocol: file
    config:
      version: 1
      storageDriverName: ontap-nas
      debug: false
      debugTraceFlags: null
      disableDelete: false
      serialNumbers:
        - nwkvzfanek_SN
      limitVolumeSize: ""
    state: online
    online: true
```

- As mensagens do AutoSupport são enviadas para o ponto de extremidade do AutoSupport do NetApp. Se você estiver usando um Registro privado para armazenar imagens de contentor, você pode usar o `--image-registry` sinalizador.
- Você também pode configurar URLs de proxy gerando os arquivos YAML de instalação. Isso pode ser feito usando `tridentctl install --generate-custom-yaml` para criar os arquivos YAML e adicionar o `--proxy-url` argumento para o `trident-autosupport` contentor no `trident-deployment.yaml`.

Desativar métricas do Trident

Para **desabilitar métricas** de serem reportadas, você deve gerar YAMLs personalizados (usando o `--generate-custom-yaml` sinalizador) e editá-los para remover o `--metrics` sinalizador de ser invocado para o `trident-main` contentor.

Desinstale o Trident

Você deve usar o mesmo método para desinstalar o Trident que você usou para instalar o Trident.

Sobre esta tarefa

- Se você precisar de uma correção para bugs observados após uma atualização, problemas de dependência ou uma atualização mal sucedida ou incompleta, você deve desinstalar o Trident e reinstalar

a versão anterior usando as instruções específicas para esse "versão". Esta é a única maneira recomendada de *downgrade* para uma versão anterior.

- Para facilitar a atualização e reinstalação, desinstalar o Trident não remove os CRDs ou objetos relacionados criados pelo Trident. Se você precisar remover completamente o Trident e todos os seus dados, "Remova completamente Trident e CRDs"consulte .

Antes de começar

Se você estiver desativando clusters do Kubernetes, exclua todas as aplicações que usam volumes criados pelo Trident antes da desinstalação. Isso garante que os PVCs sejam inéditos nos nós do Kubernetes antes que sejam excluídos.

Determine o método de instalação original

Você deve usar o mesmo método para desinstalar o Trident que você usou para instalá-lo. Antes de desinstalar, verifique qual versão você usou para instalar o Trident originalmente.

1. Use `kubectl get pods -n trident` para examinar os pods.
   - Se não houver nenhum pod do operador, o Trident foi instalado usando `tridentctl`o .
   - Se houver um pod do operador, o Trident foi instalado usando o operador Trident manualmente ou usando o Helm.
2. Se houver um pod do operador, use `kubectl describe tproc trident` para determinar se o Trident foi instalado usando o Helm.
   - Se houver uma etiqueta Helm, o Trident foi instalado usando Helm.
   - Se não houver nenhuma etiqueta Helm, o Trident foi instalado manualmente usando o operador Trident.

Desinstale a instalação de um operador Trident

Você pode desinstalar manualmente uma instalação do operador do Trident ou usando o Helm.

Desinstalar a instalação manual

Se você instalou o Trident usando o operador, você pode desinstalá-lo fazendo um dos seguintes procedimentos:

1. **Editar `TridentOrchestrator` CR e definir o sinalizador de desinstalação**:

   ```
   kubectl patch torc <trident-orchestrator-name> --type=merge -p
   '{"spec":{"uninstall":true}}'
   ```

   Quando o `uninstall` sinalizador está definido como `true`, o operador Trident desinstala o Trident, mas não remove o próprio TridentOrchestrator. Você deve limpar o TridentOrchestrator e criar um novo se quiser instalar o Trident novamente.

2. **Excluir `TridentOrchestrator`**: Ao remover o `TridentOrchestrator` CR que foi usado para implantar o Trident, você instrui o operador a desinstalar o Trident. O operador processa a remoção `TridentOrchestrator` e remove a implantação do Trident e o daemonset, excluindo os pods do Trident que ele criou como parte da instalação.

```
kubectl delete -f deploy/<bundle.yaml> -n <namespace>
```

Desinstale a instalação do Helm

Se você instalou o Trident usando o Helm, você pode desinstalá-lo usando `helm uninstall`o .

```
#List the Helm release corresponding to the Trident install.
helm ls -n trident
NAME          NAMESPACE       REVISION        UPDATED
STATUS          CHART                           APP VERSION
trident       trident         1               2021-04-20
00:26:42.417764794 +0000 UTC deployed        trident-operator-21.07.1
21.07.1

#Uninstall Helm release to remove Trident
helm uninstall trident -n trident
release "trident" uninstalled
```

Desinstale uma `tridentctl` instalação

Use o `uninstall` comando in `tridentctl` para remover todos os recursos associados ao Trident, exceto para CRDs e objetos relacionados:

```
./tridentctl uninstall -n <namespace>
```

Trident para Docker

Pré-requisitos para implantação

Você precisa instalar e configurar os pré-requisitos de protocolo necessários no seu host antes de implantar o Trident.

Verifique os requisitos

- Verifique se sua implantação atende a todos "requisitos"os .
- Verifique se você tem uma versão suportada do Docker instalada. Se a versão do Docker estiver desatualizada, "instale ou atualize-o.".

```
docker --version
```

- Verifique se os pré-requisitos do protocolo estão instalados e configurados no seu host.

Ferramentas NFS

Instale as ferramentas NFS usando os comandos do seu sistema operacional.

RHEL 8 MAIS

```
sudo yum install -y nfs-utils
```

Ubuntu

```
sudo apt-get install -y nfs-common
```

AVISO	Reinicie seus nós de trabalho após instalar as ferramentas NFS para evitar falhas ao anexar volumes a contêineres.

Ferramentas iSCSI

Instale as ferramentas iSCSI utilizando os comandos do seu sistema operativo.

RHEL 8 MAIS

1. Instale os seguintes pacotes de sistema:

```
sudo yum install -y lsscsi iscsi-initiator-utils sg3_utils device-
mapper-multipath
```

2. Verifique se a versão iscsi-iniciador-utils é 6,2.0,874-2.el7 ou posterior:

```
rpm -q iscsi-initiator-utils
```

3. Definir a digitalização para manual:

```
sudo sed -i 's/^\(node.session.scan\).*/\1 = manual/'
/etc/iscsi/iscsid.conf
```

4. Ativar multipathing:

```
sudo mpathconf --enable --with_multipathd y --find_multipaths n
```

OBSERVAÇÃO	Certifique-se de `etc/multipath.conf` que contém `find_multipaths no` `defaults` em .

5. Certifique-se de que `iscsid` e `multipathd` estão a funcionar:

```
sudo systemctl enable --now iscsid multipathd
```

6. Ativar e iniciar `iscsi`:

```
sudo systemctl enable --now iscsi
```

Ubuntu

1. Instale os seguintes pacotes de sistema:

```
sudo apt-get install -y open-iscsi lsscsi sg3-utils multipath-tools
scsitools
```

2. Verifique se a versão Open-iscsi é 2,0.874-5ubuntu2.10 ou posterior (para bionic) ou 2,0.874-7.1ubuntu6.1 ou posterior (para focal):

```
dpkg -l open-iscsi
```

3. Definir a digitalização para manual:

```
sudo sed -i 's/^\(node.session.scan\).*/\1 = manual/'
/etc/iscsi/iscsid.conf
```

4. Ativar multipathing:

```
sudo tee /etc/multipath.conf <<-EOF
defaults {
    user_friendly_names yes
    find_multipaths no
}
EOF
sudo systemctl enable --now multipath-tools.service
sudo service multipath-tools restart
```

OBSERVAÇÃO Certifique-se de `etc/multipath.conf` que contém `find_multipaths no defaults` em .

5. Certifique-se de que `open-iscsi` e `multipath-tools` estão ativados e em execução:

```
sudo systemctl status multipath-tools
sudo systemctl enable --now open-iscsi.service
sudo systemctl status open-iscsi
```

Ferramentas NVMe

Instale as ferramentas NVMe usando os comandos do seu sistema operacional.

OBSERVAÇÃO

- O NVMe requer o RHEL 9 ou posterior.
- Se a versão do kernel do seu nó Kubernetes for muito antiga ou se o pacote NVMe não estiver disponível para a versão do kernel, talvez seja necessário atualizar a versão do kernel do nó para uma com o pacote NVMe.

RHEL 9

```
sudo yum install nvme-cli
sudo yum install linux-modules-extra-$(uname -r)
sudo modprobe nvme-tcp
```

Ubuntu

```
sudo apt install nvme-cli
sudo apt -y install linux-modules-extra-$(uname -r)
sudo modprobe nvme-tcp
```

Ferramentas FC

Instale as ferramentas FC usando os comandos do seu sistema operacional.

- Ao usar nós de trabalho que executam RHEL/Red Hat Enterprise Linux CoreOS (RHCOS) com FC PVs, especifique a `discard` mountOption no StorageClass para executar a recuperação de espaço em linha. Consulte a "Documentação da Red Hat".

RHEL 8 MAIS

1. Instale os seguintes pacotes de sistema:

```
sudo yum install -y lsscsi device-mapper-multipath
```

2. Ativar multipathing:

```
sudo mpathconf --enable --with_multipathd y --find_multipaths n
```

OBSERVAÇÃO Certifique-se de `etc/multipath.conf` que contém `find_multipaths no` `defaults` em .

3. Certifique-se de que `multipathd` está em execução:

```
sudo systemctl enable --now multipathd
```

Ubuntu

1. Instale os seguintes pacotes de sistema:

```
sudo apt-get install -y lsscsi sg3-utils multipath-tools scsitools
```

2. Ativar multipathing:

```
sudo tee /etc/multipath.conf <<-EOF
defaults {
    user_friendly_names yes
    find_multipaths no
}
EOF
sudo systemctl enable --now multipath-tools.service
sudo service multipath-tools restart
```

OBSERVAÇÃO Certifique-se de `etc/multipath.conf` que contém `find_multipaths`

`no defaults` em .

3. Certifique-se de que `multipath-tools` está ativado e em execução:

```
sudo systemctl status multipath-tools
```

Implante o Trident

O Trident para Docker oferece integração direta com o ecossistema Docker para plataformas de storage NetApp. Ele dá suporte ao provisionamento e gerenciamento de recursos de storage da plataforma de storage para hosts Docker, com uma estrutura para adicionar plataformas adicionais no futuro.

Várias instâncias do Trident podem ser executadas simultaneamente no mesmo host. Isso permite conexões simultâneas a vários sistemas de armazenamento e tipos de armazenamento, com a capacidade de personalizar o armazenamento usado para os volumes Docker.

O que você vai precisar

Consulte "pré-requisitos para implantação". Depois de garantir que os pré-requisitos sejam atendidos, você estará pronto para implantar o Trident.

Método de plug-in gerenciado Docker (versão 1,13/17,03 e posterior)

OBSERVAÇÃO	**Antes de começar** Se você usou o Trident pré Docker 1,13/17,03 no método daemon tradicional, certifique-se de parar o processo Trident e reiniciar o seu daemon Docker antes de usar o método do plugin gerenciado.

1. Parar todas as instâncias em execução:

```
pkill /usr/local/bin/netappdvp
pkill /usr/local/bin/trident
```

2. Reinicie o Docker.

```
systemctl restart docker
```

3. Certifique-se de que tem o Docker Engine 17,03 (novo 1,13) ou posterior instalado.

```
docker --version
```

Se a sua versão estiver desatualizada, "instale ou atualize a instalação".

Passos

1. Crie um arquivo de configuração e especifique as opções da seguinte forma:
   - `config`: O nome do arquivo padrão é `config.json`, no entanto, você pode usar qualquer nome que você escolher especificando a `config` opção com o nome do arquivo. O arquivo de configuração deve estar localizado `/etc/netappdvp` no diretório no sistema host.
   - `log-level`: Especifique o nível de registo (`debug`, `info`, `warn` `error` , , `fatal`). A predefinição é `info`.
   - `debug`: Especifique se o log de depuração está ativado. O padrão é falso. Substitui o nível de log, se verdadeiro.
     i. Crie um local para o arquivo de configuração:

     ```
     sudo mkdir -p /etc/netappdvp
     ```

     ii. Crie o arquivo de configuração:

     ```
     cat << EOF > /etc/netappdvp/config.json
     ```

     ```
     {
       "version": 1,
       "storageDriverName": "ontap-nas",
       "managementLIF": "10.0.0.1",
       "dataLIF": "10.0.0.2",
       "svm": "svm_nfs",
       "username": "vsadmin",
       "password": "password",
       "aggregate": "aggr1"
     }
     EOF
     ```

2. Inicie o Trident usando o sistema de plug-in gerenciado. Substitua `<version>` pela versão do plugin (xxx.xx.x) que você está usando.

   ```
   docker plugin install --grant-all-permissions --alias netapp
   netapp/trident-plugin:<version> config=myConfigFile.json
   ```

3. Comece a usar o Trident para consumir storage do sistema configurado.
   a. Crie um volume chamado "firstvolume":

   ```
   docker volume create -d netapp --name firstVolume
   ```

b. Crie um volume padrão quando o contentor for iniciado:

```
docker run --rm -it --volume-driver netapp --volume
secondVolume:/my_vol alpine ash
```

c. Remover o volume "firstvolume":

```
docker volume rm firstVolume
```

Método tradicional (versão 1,12 ou anterior)

Antes de começar

1. Certifique-se de que você tem o Docker versão 1,10 ou posterior.

```
docker --version
```

Se a sua versão estiver desatualizada, atualize a instalação.

```
curl -fsSL https://get.docker.com/ | sh
```

Ou, "siga as instruções para sua distribuição".

2. Certifique-se de que NFS e/ou iSCSI estão configurados para o seu sistema.

Passos

1. Instale e configure o plug-in de volume do Docker do NetApp:

a. Baixe e descompacte o aplicativo:

```
----trident-installer-26.10.1.tar.gz
wget https://github.com/NetApp/trident/releases/download/10.0/
tar zxf trident-installer-26.10.1.tar.gz
```

```
.. Move to a location in the bin path:
+
[source,console]
```

sudo mv trident-installer/extras/bin/trident /usr/local/bin/ sudo chown root:root /usr/local/bin/trident sudo chmod 755 /usr/local/bin/trident

```
.. Create a location for the configuration file:
+
```

```
[source,console]
```

sudo mkdir -p /etc/netappdvp

```
.. Create the configuration file:
+
[source,console]
```

cat << EOF > /etc/netappdvp/ontap-nas.json

```
+
[source,json]
```

{ "version": 1, "storageDriverName": "ontap-nas", "managementLIF": "10.0.0.1", "dataLIF": "10.0.0.2", "svm": "svm_nfs", "username": "vsadmin", "password": "password", "aggregate": "aggr1" } EOF

```
. After placing the binary and creating the configuration file, start the
Trident daemon using the desired configuration file.
+
[source,console]
```

sudo trident --config=/etc/netappdvp/ontap-nas.json

```
+
NOTE: Unless specified, the default name for the volume driver is
"netapp".
+
After the daemon is started, you can create and manage volumes by using
the Docker CLI interface.

. Create a volume:
+
[source,console]
```

docker volume create -d netapp --name trident_1

```
. Provision a Docker volume when starting a container:
+
[source,console]
```

docker run --rm -it --volume-driver netapp --volume trident_2:/my_vol alpine ash

```
. Remove a Docker volume:
+
[source,console]
```

docker volume rm trident_1

```
+
[source,console]
```

docker volume rm trident_2

```
== Start Trident at system startup

A sample unit file for systemd based systems can be found at
`contrib/trident.service.example` in the Git repo. To use the file with
RHEL, do the following:

. Copy the file to the correct location.
+
You should use unique names for the unit files if you have more than one
instance running.
+
[source,console]
```

cp contrib/trident.service.example /usr/lib/systemd/system/trident.service

```
. Edit the file, change the description (line 2) to match the driver name
and the configuration file path (line 9) to reflect your environment.
. Reload systemd for it to ingest changes:
+
[source,console]
```

systemctl daemon-reload

```
. Enable the service.
+
This name varies depending on what you named the file in the
`/usr/lib/systemd/system` directory.
+
[source,console]
```

systemctl enable trident

```
. Start the service.
+
[source,console]
```

systemctl start trident

```
. View the status.
+
[source,console]
```

systemctl status trident

```
NOTE: Any time you modify the unit file, run the `systemctl daemon-reload`
command for it to be aware of the changes.
```

Atualize ou desinstale o Trident

Você pode atualizar com segurança o Trident para Docker sem qualquer impactos nos volumes que estão em uso. Durante o processo de atualização, haverá um breve período em que `docker volume` os comandos direcionados para o plugin não serão bem-sucedidos, e os aplicativos não poderão montar volumes até que o plugin esteja sendo executado novamente. Na maioria das circunstâncias, esta é uma questão de segundos.

Atualização

Execute as etapas abaixo para atualizar o Trident para Docker.

Passos

1. Listar os volumes existentes:

```
docker volume ls
DRIVER              VOLUME NAME
netapp:latest       my_volume
```

2. Desativar o plugin:

```
docker plugin disable -f netapp:latest
docker plugin ls
ID                  NAME                DESCRIPTION
```

```
ENABLED
7067f39a5df5        netapp:latest       nDVP - NetApp Docker Volume
Plugin   false
```

3. Atualize o plugin:

```
docker plugin upgrade --skip-remote-check --grant-all-permissions
netapp:latest netapp/trident-plugin:21.07
```

OBSERVAÇÃO A versão 18,01 do Trident substitui o nDVP. Você deve atualizar diretamente da `netapp/ndvp-plugin` imagem para a `netapp/trident-plugin` imagem.

4. Ativar o plugin:

```
docker plugin enable netapp:latest
```

5. Verifique se o plugin está ativado:

```
docker plugin ls
ID                  NAME                DESCRIPTION
ENABLED
7067f39a5df5        netapp:latest       Trident - NetApp Docker Volume
Plugin   true
```

6. Verifique se os volumes estão visíveis:

```
docker volume ls
DRIVER              VOLUME NAME
netapp:latest       my_volume
```

IMPORTANTE Se você estiver atualizando de uma versão antiga do Trident (pré-20,10) para o Trident 20,10 ou posterior, você pode encontrar um erro. Para obter mais informações, "Problemas conhecidos"consulte . Se você correr para o erro, você deve primeiro desativar o plugin, em seguida, remover o plugin e, em seguida, instalar a versão necessária do Trident passando um parâmetro de configuração extra: `docker plugin install netapp/trident-plugin:20.10 --alias netapp --grant-all-permissions config=config.json`

Desinstalar

Execute as etapas abaixo para desinstalar o Trident para Docker.

Passos

1. Remova todos os volumes criados pelo plugin.
2. Desativar o plugin:

```
docker plugin disable netapp:latest
docker plugin ls
ID                  NAME                DESCRIPTION
ENABLED
7067f39a5df5        netapp:latest       nDVP - NetApp Docker Volume
Plugin   false
```

3. Remova o plugin:

```
docker plugin rm netapp:latest
```

Trabalhe com volumes

Você pode criar, clonar e remover volumes facilmente usando os comandos padrão `docker volume` com o nome do driver Trident especificado quando necessário.

Crie um volume

- Crie um volume com um driver usando o nome padrão:

```
docker volume create -d netapp --name firstVolume
```

- Crie um volume com uma instância do Trident específica:

```
docker volume create -d ntap_bronze --name bronzeVolume
```

OBSERVAÇÃO Se você não especificar nenhum "opções", os padrões para o driver serão usados.

- Substituir o tamanho padrão do volume. Veja o exemplo a seguir para criar um volume de 20 GiB com um driver:

```
docker volume create -d netapp --name my_vol --opt size=20G
```

DICA Os tamanhos de volume são expressos como strings contendo um valor inteiro com unidades opcionais (exemplo: 10g, 20GB, 3TiB). Se nenhuma unidade for especificada, o padrão é G. as unidades de tamanho podem ser expressas como potências de 2 (B, KiB, MIB, GiB, TIB) ou potências de 10 (B, KB, MB, GB, TB). As unidades shorthand usam poderes de 2 (G GiB, T TIB,…).

Remova um volume

- Remova o volume como qualquer outro volume do Docker:

```
docker volume rm firstVolume
```

IMPORTANTE	Ao utilizar o `solidfire-san` controlador, o exemplo acima elimina e elimina o volume.

Execute as etapas abaixo para atualizar o Trident para Docker.

Clonar um volume

Ao usar o `ontap-nas` , `ontap-san` , e `solidfire-san` Os drivers de armazenamento, como o Trident, podem clonar volumes. Ao usar o `ontap-nas-flexgroup` ou `ontap-nas-economy` Drivers, clonagem não é suportada. Criar um novo volume a partir de um volume existente resultará na criação de um novo snapshot.

- Inspecione o volume para enumerar instantâneos:

```
docker volume inspect <volume_name>
```

- Crie um novo volume a partir de um volume existente. Isso resultará na criação de um novo snapshot:

```
docker volume create -d <driver_name> --name <new_name> -o from
=<source_docker_volume>
```

- Criar um novo volume a partir de um instantâneo existente em um volume. Isso não criará um novo snapshot:

```
docker volume create -d <driver_name> --name <new_name> -o from
=<source_docker_volume> -o fromSnapshot=<source_snap_name>
```

Exemplo

```
docker volume inspect firstVolume

[
  {
    "Driver": "ontap-nas",
    "Labels": null,
    "Mountpoint": "/var/lib/docker-volumes/ontap-
nas/netappdvp_firstVolume",
    "Name": "firstVolume",
    "Options": {},
```

```
    "Scope": "global",
    "Status": {
      "Snapshots": [
        {
          "Created": "2017-02-10T19:05:00Z",
          "Name": "hourly.2017-02-10_1505"
        }
      ]
    }
  }
]

docker volume create -d ontap-nas --name clonedVolume -o from=firstVolume
clonedVolume

docker volume rm clonedVolume
docker volume create -d ontap-nas --name volFromSnap -o from=firstVolume
-o fromSnapshot=hourly.2017-02-10_1505
volFromSnap

docker volume rm volFromSnap
```

Acesse volumes criados externamente

Você pode acessar dispositivos de bloco criados externamente (ou seus clones) por contentores usando Trident **only** se eles não tiverem partições e se seu sistema de arquivos for suportado pelo Trident (por exemplo: Um `ext4`-formatado `/dev/sdc1` não será acessível via Trident).

Opções de volume específicas do condutor

Cada driver de armazenamento tem um conjunto diferente de opções, que você pode especificar no momento da criação do volume para personalizar o resultado. Veja abaixo as opções que se aplicam ao sistema de armazenamento configurado.

Usar essas opções durante a operação de criação de volume é simples. Forneça a opção e o valor usando o `-o` operador durante a operação CLI. Estes substituem quaisquer valores equivalentes do arquivo de configuração JSON.

Opções de volume ONTAP

As opções de criação de volume para NFS, iSCSI e FC incluem o seguinte:

Opção	Descrição
`size`	O tamanho do volume, padrão é 1 GiB.
`spaceReserve`	Provisionamento fino ou espesso do volume, o padrão é fino. Os valores válidos são `none` (thin Provisioning) e `volume` (thick provisioned).

Opção	Descrição
`snapshotPolicy`	Isto irá definir a política de instantâneos para o valor pretendido. O padrão é `none`, o que significa que nenhum instantâneo será criado automaticamente para o volume. A menos que seja modificada pelo administrador de storage, existe uma política chamada "padrão" em todos os sistemas ONTAP, que cria e retém seis snapshots por hora, dois por dia e dois por semana. Os dados preservados em um snapshot podem ser recuperados navegando para `.snapshot` o diretório em qualquer diretório do volume.
`snapshotReserve`	Isto irá definir a reserva de instantâneos para a percentagem pretendida. O padrão não é nenhum valor, o que significa que o ONTAP selecionará o snapshotServe (geralmente 5%) se você selecionou uma política de snapshotPolicy, ou 0% se a política de snapshotPolicy não for nenhuma. Você pode definir o valor padrão snapshotServe no arquivo de configuração para todos os backends ONTAP, e você pode usá-lo como uma opção de criação de volume para todos os backends ONTAP, exceto ONTAP-nas-economy.
`splitOnClone`	Ao clonar um volume, isso fará com que o ONTAP divida imediatamente o clone de seu pai. A predefinição é `false`. Alguns casos de uso para clonagem de volumes são melhor servidos dividindo o clone de seu pai imediatamente após a criação, porque é improvável que haja alguma oportunidade de eficiência de storage. Por exemplo, clonar um banco de dados vazio pode oferecer grande economia de tempo, mas pouca economia de armazenamento, por isso é melhor dividir o clone imediatamente.
`encryption`	Ative a criptografia de volume do NetApp (NVE) no novo volume; o padrão é `false`. O NVE deve ser licenciado e habilitado no cluster para usar essa opção. Se NAE estiver ativado no back-end, qualquer volume provisionado no Trident será NAE habilitado. Para obter mais informações, consulte: "Como o Trident funciona com NVE e NAE".
`tieringPolicy`	Define a política de disposição em categorias a ser usada para o volume. Isso decide se os dados são movidos para a categoria de nuvem quando ficam inativos (frios).

As seguintes opções adicionais são para NFS **somente**:

Opção	Descrição
`unixPermissions`	Isso controla o conjunto de permissões para o próprio volume. Por padrão, as permissões serão definidas como `---rwxr-xr-x, ou em notação numérica 0755, e `root` serão o proprietário. O texto ou o formato numérico funcionarão.
`snapshotDir`	Definir isso como `true` tornará o `.snapshot` diretório visível para os clientes que acessam o volume. O valor padrão é `false`, o que significa que a visibilidade `.snapshot` do diretório está desativada por padrão. Algumas imagens, por exemplo, a imagem oficial do MySQL, não funcionam como esperado quando o `.snapshot` diretório está visível.
`exportPolicy`	Define a política de exportação a ser utilizada para o volume. A predefinição é `default`.
`securityStyle`	Define o estilo de segurança a ser usado para acesso ao volume. A predefinição é `unix`. Os valores válidos são `unix` e `mixed`.

As seguintes opções adicionais são para iSCSI **somente**:

Opção	Descrição
`fileSystemType`	Define o sistema de ficheiros utilizado para formatar volumes iSCSI. A predefinição é `ext4`. Os valores válidos são `ext3`, `ext4`, e `xfs`.
`spaceAllocation`	Definir esta opção como `false` desativa a funcionalidade de alocação de espaço do LUN. O valor padrão é `true`, o que significa que o ONTAP notifica o host quando o volume ficou sem espaço e o LUN no volume não pode aceitar gravações. Essa opção também permite que o ONTAP recupere espaço automaticamente quando o host exclui dados.

Exemplos

Veja os exemplos abaixo:

- Crie um volume de 10 GiB:

```
docker volume create -d netapp --name demo -o size=10G -o
encryption=true
```

- Crie um volume de 100 GiB com snapshots:

```
docker volume create -d netapp --name demo -o size=100G -o
snapshotPolicy=default -o snapshotReserve=10
```

- Crie um volume que tenha o bit setuid ativado:

```
docker volume create -d netapp --name demo -o unixPermissions=4755
```

O tamanho mínimo do volume é 20 MiB.

Se a reserva de snapshot não for especificada e a política de snapshot for `none`, o Trident usará uma reserva de snapshot de 0%.

- Criar um volume sem política de snapshot e sem reserva de snapshot:

```
docker volume create -d netapp --name my_vol --opt snapshotPolicy=none
```

- Crie um volume sem política de snapshot e uma reserva de snapshot personalizada de 10%:

```
docker volume create -d netapp --name my_vol --opt snapshotPolicy=none
--opt snapshotReserve=10
```

- Crie um volume com uma política de snapshot e uma reserva de snapshot personalizada de 10%:

```
docker volume create -d netapp --name my_vol --opt
snapshotPolicy=myPolicy --opt snapshotReserve=10
```

- Crie um volume com uma política de snapshot e aceite a reserva de snapshot padrão do ONTAP (geralmente 5%):

```
docker volume create -d netapp --name my_vol --opt
snapshotPolicy=myPolicy
```

Opções de volume do software Element

As opções de software Element expõem as políticas de tamanho e qualidade do serviço (QoS) associadas ao volume. Quando o volume é criado, a política de QoS associada a ele é especificada usando a `-o type=service_level` nomenclatura.

A primeira etapa para definir um nível de serviço QoS com o driver Element é criar pelo menos um tipo e especificar o IOPS mínimo, máximo e de pico associado a um nome no arquivo de configuração.

Outras opções de criação de volume de software Element incluem o seguinte:

Opção	Descrição
`size`	O tamanho do volume, por padrão, é 1 GiB ou entrada de configuração … "defaults": {"size": "5G"}.

Opção	Descrição
`blocksize`	Use 512 ou 4096, o padrão é 512 ou a entrada de configuração DefaultBlockSize.

Exemplo

Veja o seguinte arquivo de configuração de exemplo com definições de QoS:

```
{
  "Types": [
    {
      "Type": "Bronze",
      "Qos": {
        "minIOPS": 1000,
        "maxIOPS": 2000,
        "burstIOPS": 4000
      }
    },
    {
      "Type": "Silver",
      "Qos": {
        "minIOPS": 4000,
        "maxIOPS": 6000,
        "burstIOPS": 8000
      }
    },
    {
      "Type": "Gold",
      "Qos": {
        "minIOPS": 6000,
        "maxIOPS": 8000,
        "burstIOPS": 10000
      }
    }
  ]
}
```

Na configuração acima, temos três definições de política: Bronze, prata e ouro. Esses nomes são arbitrários.

- Crie um volume Gold de 10 GiB:

```
docker volume create -d solidfire --name sfGold -o type=Gold -o size=10G
```

- Crie um volume Bronze de 100 GiB:

```
docker volume create -d solidfire --name sfBronze -o type=Bronze -o
size=100G
```

Recolher registos

Você pode coletar Registros para obter ajuda com a solução de problemas. O método que você usa para coletar os logs varia de acordo com a forma como você está executando o plugin Docker.

Recolha registos para resolução de problemas

Passos

1. Se você estiver executando o Trident usando o método recomendado de plug-in gerenciado (ou seja, usando `docker plugin` comandos), visualize-os da seguinte forma:

```
docker plugin ls
```

```
ID                  NAME                DESCRIPTION
ENABLED
4fb97d2b956b        netapp:latest       nDVP - NetApp Docker Volume
Plugin   false
journalctl -u docker | grep 4fb97d2b956b
```

O nível de registo padrão deve permitir diagnosticar a maioria dos problemas. Se você achar que isso não é suficiente, você pode ativar o Registro de depuração.

2. Para ativar o registo de depuração, instale o plug-in com o registo de depuração ativado:

```
docker plugin install netapp/trident-plugin:<version> --alias <alias>
debug=true
```

Ou, ative o registo de depuração quando o plug-in já estiver instalado:

```
docker plugin disable <plugin>
```

```
docker plugin set <plugin> debug=true
```

```
docker plugin enable <plugin>
```

3. Se você estiver executando o binário em si no host, os logs estarão disponíveis no diretório do host `/var/log/netappdvp`. Para ativar o registo de depuração, especifique `-debug` quando executar o plugin.

Dicas gerais de solução de problemas

- O problema mais comum em que novos usuários são executados é uma configuração incorreta que impede que o plugin seja inicializado. Quando isso acontecer, você provavelmente verá uma mensagem como esta quando você tentar instalar ou ativar o plugin:

  ```
  Error response from daemon: dial unix /run/docker/plugins/<id>/netapp.sock:
  connect: no such file or directory
  ```

  Isto significa que o plugin falhou ao iniciar. Felizmente, o plugin foi construído com uma capacidade de Registro abrangente que deve ajudá-lo a diagnosticar a maioria dos problemas que você provavelmente encontrará.

- Se houver problemas com a montagem de um PV em um recipiente, certifique-se de que `rpcbind` está instalado e funcionando. Use o gerenciador de pacotes necessário para o sistema operacional do host e verifique se `rpcbind` está em execução. Você pode verificar o status do serviço rpcbind executando um `systemctl status rpcbind` ou seu equivalente.

Gerenciar várias instâncias do Trident

Várias instâncias do Trident são necessárias quando você deseja ter várias configurações de storage disponíveis simultaneamente. A chave para várias instâncias é dar nomes diferentes usando a `--alias` opção com o plug-in em contentor, ou `--volume-driver` opção ao instanciar o Trident no host.

Etapas para o plugin gerenciado do Docker (versão 1,13/17,03 ou posterior)

1. Inicie a primeira instância especificando um alias e um arquivo de configuração.

   ```
   docker plugin install --grant-all-permissions --alias silver
   netapp/trident-plugin:21.07 config=silver.json
   ```

2. Inicie a segunda instância, especificando um alias diferente e arquivo de configuração.

   ```
   docker plugin install --grant-all-permissions --alias gold
   netapp/trident-plugin:21.07 config=gold.json
   ```

3. Crie volumes especificando o alias como o nome do driver.

   Por exemplo, para o volume de ouro:

   ```
   docker volume create -d gold --name ntapGold
   ```

Por exemplo, para o volume prateado:

```
docker volume create -d silver --name ntapSilver
```

Passos para o tradicional (versão 1,12 ou anterior)

1. Inicie o plugin com uma configuração NFS usando um ID de driver personalizado:

```
sudo trident --volume-driver=netapp-nas --config=/path/to/config
-nfs.json
```

2. Inicie o plug-in com uma configuração iSCSI usando um ID de driver personalizado:

```
sudo trident --volume-driver=netapp-san --config=/path/to/config
-iscsi.json
```

3. Provisione volumes Docker para cada instância de driver:

   Por exemplo, para NFS:

```
docker volume create -d netapp-nas --name my_nfs_vol
```

   Por exemplo, para iSCSI:

```
docker volume create -d netapp-san --name my_iscsi_vol
```

Opções de configuração de armazenamento

Consulte as opções de configuração disponíveis para as configurações do Trident.

Opções de configuração global

Essas opções de configuração se aplicam a todas as configurações do Trident, independentemente da plataforma de storage usada.

Opção	Descrição	Exemplo
`version`	Número da versão do ficheiro de configuração	`1`
`storageDriverName`	Nome do driver de armazenamento	`ontap-nas` `ontap-san`, , `ontap-nas-economy` `ontap-nas-flexgroup` , , , `solidfire-san`

Opção	Descrição	Exemplo
`storagePrefix`	Prefixo opcional para nomes de volume. Padrão: `netappdvp_`.	`staging_`
`limitVolumeSize`	Restrição opcional nos tamanhos de volume. Padrão: "" (não aplicado)	`10g`

DICA Não use `storagePrefix` (incluindo o padrão) para backends de elemento. Por padrão, o `solidfire-san` driver ignorará essa configuração e não usará um prefixo. O NetApp recomenda usar um tenantID específico para mapeamento de volume do Docker ou usar os dados de atributo que são preenchidos com a versão do Docker, informações de driver e nome bruto do Docker nos casos em que qualquer nome munging pode ter sido usado.

As opções padrão estão disponíveis para evitar ter que especificá-las em cada volume criado. A `size` opção está disponível para todos os tipos de controlador. Consulte a seção Configuração do ONTAP para obter um exemplo de como definir o tamanho padrão do volume.

Opção	Descrição	Exemplo
`size`	Tamanho padrão opcional para novos volumes. Predefinição: `1G`	`10G`

Configuração ONTAP

Além dos valores de configuração global acima, ao usar o ONTAP, as seguintes opções de nível superior estão disponíveis.

Opção	Descrição	Exemplo
`managementLIF`	Endereço IP do ONTAP Management LIF. Você pode especificar um nome de domínio totalmente qualificado (FQDN).	`10.0.0.1`

Opção	Descrição	Exemplo
`dataLIF`	Endereço IP do protocolo LIF. • ONTAP nas drivers*: A NetApp recomenda especificar `dataLIF`. Se não for fornecido, o Trident buscará os dados LIFs do SVM. Você pode especificar um nome de domínio totalmente qualificado (FQDN) a ser usado para as operações de montagem NFS, permitindo que você crie um DNS round-robin para balanceamento de carga entre vários dataLIFs. **Drivers SAN ONTAP**: Não especifique para iSCSI ou FC. O Trident usa "Mapa de LUN seletivo da ONTAP" para descobrir as LIFs iSCSI ou FC necessárias para estabelecer uma sessão de vários caminhos. Um aviso é gerado se `dataLIF` for definido explicitamente.	`10.0.0.2`
`svm`	Máquina virtual de armazenamento a utilizar (necessária, se o LIF de gestão for um LIF de cluster)	`svm_nfs`
`username`	Nome de utilizador para ligar ao dispositivo de armazenamento	`vsadmin`
`password`	Palavra-passe para ligar ao dispositivo de armazenamento	`secret`
`aggregate`	Agregado para provisionamento (opcional; se definido, deve ser atribuído ao SVM). Para `ontap-nas-flexgroup` o driver, essa opção é ignorada. Todos os agregados atribuídos ao SVM são usados para provisionar um volume FlexGroup.	`aggr1`
`limitAggregateUsage`	Opcional, falha no provisionamento se o uso estiver acima dessa porcentagem	`75%`

Opção	Descrição	Exemplo
`nfsMountOptions`	Controle refinado das opções de montagem NFS; o padrão é "-o nfsvers 3". **Disponível apenas para os `ontap-nas` condutores e `ontap-nas-economy`.** "Consulte as informações de configuração do host NFS aqui".	`-o nfsvers=4`
`igroupName`	O Trident cria e gerencia por nó `igroups` `netappdvp` como . Este valor não pode ser alterado ou omitido. **Disponível apenas para `ontap-san` o condutor.**	`netappdvp`
`limitVolumeSize`	Tamanho máximo do volume requestable.	`300g`
`qtreesPerFlexvol`	Qtrees máximos por FlexVol, tem de estar no intervalo [50, 300], o padrão é 200. **Para `ontap-nas-economy` o driver, esta opção permite personalizar o número máximo de qtrees por FlexVol.**	`300`
`sanType`	**Suportado apenas para `ontap-san` driver.** Use para selecionar `iscsi` iSCSI, `nvme` NVMe/TCP ou `fcp` SCSI por Fibre Channel (FC).	`iscsi` se estiver em branco
`limitVolumePoolSize`	**Suportado apenas para `ontap-san-economy` drivers e `ontap-san-economy`.** Limites tamanhos de FlexVol em motoristas econômicos ONTAP ONTAP-nas-Economy e ONTAP-SAN-Economy.	`300g`

As opções padrão estão disponíveis para evitar ter que especificá-las em cada volume criado:

Opção	Descrição	Exemplo
`spaceReserve`	Modo de reserva de espaço; `none` (thin Provisioning) ou `volume` (thick)	`none`
`snapshotPolicy`	Política de instantâneos a utilizar, a predefinição é `none`	`none`
`snapshotReserve`	Percentagem de reserva de instantâneo, o padrão é "" para aceitar o padrão ONTAP	`10`

Opção	Descrição	Exemplo
`splitOnClone`	Divida um clone de seu pai na criação, o padrão é `false`	`false`
`encryption`	Ativa a criptografia de volume NetApp (NVE) no novo volume; o padrão é `false`. O NVE deve ser licenciado e habilitado no cluster para usar essa opção. Se NAE estiver ativado no back-end, qualquer volume provisionado no Trident será NAE habilitado. Para obter mais informações, consulte: "Como o Trident funciona com NVE e NAE".	verdadeiro
`unixPermissions`	Opção nas para volumes NFS provisionados, o padrão é `777`	`777`
`snapshotDir`	Opção nas para acesso ao `.snapshot` diretório.	Habilita ou desabilita o acesso ao `.snapshot` diretório. Valores válidos: `true`, `false`. O comportamento padrão pode variar de acordo com a versão do ONTAP e do Trident. Defina explicitamente para garantir um comportamento consistente.
`exportPolicy`	A opção nas para a política de exportação NFS a usar, o padrão é `default`	`default`
`securityStyle`	Opção nas para acesso ao volume NFS provisionado. Estilos de segurança e `unix` suporte de NFS `mixed`. A predefinição é `unix`.	`unix`
`fileSystemType`	Opção SAN para selecionar o tipo de sistema de arquivos, o padrão é `ext4`	`xfs`
`tieringPolicy`	A política de disposição em categorias a ser usada, o padrão é `none`.	`none`
`skipRecoveryQueue`	Durante a exclusão de um volume, ignore a fila de recuperação no armazenamento e exclua o volume imediatamente.	``

Opções de dimensionamento

Os `ontap-nas` drivers e `ontap-san` criam um ONTAP FlexVol para cada volume do Docker. O ONTAP dá suporte a até 1000 FlexVols por nó de cluster com um cluster máximo de 12.000 volumes FlexVol se os requisitos de volume do Docker se ajustarem a essa limitação, `ontap-nas` o driver é a solução nas preferida devido aos recursos adicionais oferecidos pelo FlexVols, como snapshots Docker granular e clonagem.

Se você precisar de mais volumes do Docker do que pode ser acomodado pelos limites do FlexVol, escolha o `ontap-nas-economy` ou o `ontap-san-economy` driver.

`ontap-nas-economy`O driver cria volumes do Docker como Qtrees do ONTAP em um pool de volumes do FlexVol gerenciados automaticamente. As Qtrees oferecem dimensionamento muito maior, até 100.000 PB por nó de cluster e 2.400.000 PB por cluster, à custa de alguns recursos. `ontap-nas-economy`O driver não oferece suporte a snapshots ou clonagem granular de volume do Docker.

OBSERVAÇÃO O `ontap-nas-economy` driver atualmente não é suportado no Docker Swarm, porque o Docker Swarm não orquestra criação de volume em vários nós.

`ontap-san-economy`O driver cria volumes do Docker como LUNs do ONTAP em um pool compartilhado de volumes do FlexVol gerenciados automaticamente. Dessa forma, cada FlexVol não se restringe a apenas um LUN e oferece melhor escalabilidade para workloads SAN. Dependendo do storage array, o ONTAP oferece suporte para até 16384 LUNs por cluster. Como os volumes são LUNs abaixo, esse driver oferece suporte a snapshots e clonagem granular do Docker volume.

Escolha o `ontap-nas-flexgroup` driver para aumentar o paralelismo para um único volume que pode crescer para o intervalo de petabytes com bilhões de arquivos. Alguns casos de uso ideais para FlexGroups incluem IA/ML/DL, big data e análise, compilações de software, streaming, repositórios de arquivos e assim por diante. O Trident usa todos os agregados atribuídos a uma SVM ao provisionar um volume FlexGroup. O suporte do FlexGroup no Trident também tem as seguintes considerações:

- Requer ONTAP versão 9,2 ou superior.
- A partir desta redação, FlexGroups só suportam NFS v3.
- Recomendado para ativar os identificadores NFSv3 de 64 bits para o SVM.
- O tamanho mínimo recomendado do membro/volume do FlexGroup é 100 GiB.
- A clonagem não é compatível com volumes FlexGroup.

Para obter informações sobre FlexGroups e cargas de trabalho que são apropriadas para FlexGroups, consulte o "Guia de práticas recomendadas e implementação do volume NetApp FlexGroup".

Para obter recursos avançados e grande escala no mesmo ambiente, é possível executar várias instâncias do Docker volume Plugin, com uma usando `ontap-nas` e outra usando `ontap-nas-economy`o .

Função ONTAP personalizada para Trident

Você pode criar uma função de cluster do ONTAP com Privileges mínimo para que você não precise usar a função de administrador do ONTAP para executar operações no Trident. Quando você inclui o nome de usuário em uma configuração de back-end do Trident, o Trident usa a função de cluster do ONTAP criada para executar as operações.

"Gerador de função personalizada Trident"Consulte para obter mais informações sobre como criar funções personalizadas do Trident.

Usando a CLI do ONTAP

1. Crie uma nova função usando o seguinte comando:

```
security login role create <role_name\> -cmddirname "command" -access all
-vserver <svm_name\>
```

2. Crie um nome de usuário para o usuário do Trident:

```
security login create -username <user_name\> -application ontapi
-authmethod password -role <name_of_role_in_step_1\> -vserver <svm_name\>
-comment "user_description"
security login create -username <user_name\> -application http -authmethod
password -role <name_of_role_in_step_1\> -vserver <svm_name\> -comment
"user_description"
```

3. Mapeie a função para o usuário:

```
security login modify username <user_name\> -vserver <svm_name\> -role
<role_name\> -application ontapi -application console -authmethod
<password\>
```

Usando o System Manager

Execute as seguintes etapas no Gerenciador do sistema do ONTAP:

1. **Crie uma função personalizada**:
   a. Para criar uma função personalizada no nível do cluster, selecione **Cluster > Settings**.

      (Ou) para criar uma função personalizada no nível SVM, selecione **Storage > Storage VMs > `required SVM` Settings > Users and Roles**.

   b. Selecione o ícone de seta (→) ao lado de **usuários e funções**.
   c. Selecione * Adicionar * em **funções**.
   d. Defina as regras para a função e clique em **Salvar**.
2. **Mapeie a função para o usuário do Trident**: Execute as seguintes etapas na página **usuários e funções**:
   a. Selecione Adicionar ícone *** em *usuários**.
   b. Selecione o nome de usuário desejado e selecione uma função no menu suspenso para **função**.
   c. Clique em **Salvar**.

Consulte as páginas a seguir para obter mais informações:

- "Funções personalizadas para administração do ONTAP" ou "Definir funções personalizadas"
- "Trabalhe com funções e usuários"

Exemplo de arquivos de configuração do ONTAP

▼ Exemplo de NFS para o driver <code> ONTAP-nas</code>

```
{
  "version": 1,
  "storageDriverName": "ontap-nas",
  "managementLIF": "10.0.0.1",
  "dataLIF": "10.0.0.2",
  "svm": "svm_nfs",
  "username": "vsadmin",
  "password": "password",
  "aggregate": "aggr1",
  "defaults": {
    "size": "10G",
    "spaceReserve": "none",
    "exportPolicy": "default"
  }
}
```

▼ **Exemplo de NFS para o driver <code> ONTAP-nas-FlexGroup </code>**

```
{
  "version": 1,
  "storageDriverName": "ontap-nas-flexgroup",
  "managementLIF": "10.0.0.1",
  "dataLIF": "10.0.0.2",
  "svm": "svm_nfs",
  "username": "vsadmin",
  "password": "password",
  "defaults": {
    "size": "100G",
    "spaceReserve": "none",
    "exportPolicy": "default"
  }
}
```

▼ **Exemplo de NFS para o driver <code> ONTAP-nas-economy</code>**

```
{
  "version": 1,
  "storageDriverName": "ontap-nas-economy",
  "managementLIF": "10.0.0.1",
  "dataLIF": "10.0.0.2",
  "svm": "svm_nfs",
  "username": "vsadmin",
  "password": "password",
  "aggregate": "aggr1"
```

```
}
```

▼ **Exemplo iSCSI para o controlador <code> ONTAP-san</code>**

```
{
  "version": 1,
  "storageDriverName": "ontap-san",
  "managementLIF": "10.0.0.1",
  "dataLIF": "10.0.0.3",
  "svm": "svm_iscsi",
  "username": "vsadmin",
  "password": "password",
  "aggregate": "aggr1",
  "igroupName": "netappdvp"
}
```

▼ **Exemplo de NFS para o driver <code> ONTAP-San-economy</code>**

```
{
  "version": 1,
  "storageDriverName": "ontap-san-economy",
  "managementLIF": "10.0.0.1",
  "dataLIF": "10.0.0.3",
  "svm": "svm_iscsi_eco",
  "username": "vsadmin",
  "password": "password",
  "aggregate": "aggr1",
  "igroupName": "netappdvp"
}
```

▼ **Exemplo de NVMe/TCP para o driver <code> ONTAP-san</code>**

```
{
  "version": 1,
  "backendName": "NVMeBackend",
  "storageDriverName": "ontap-san",
  "managementLIF": "10.0.0.1",
  "svm": "svm_nvme",
  "username": "vsadmin",
  "password": "password",
  "sanType": "nvme",
  "useREST": true
}
```

▼ **Exemplo de SCSI sobre FC para o driver <code> ONTAP-san</code>**

```
{
  "version": 1,
  "backendName": "ontap-san-backend",
  "storageDriverName": "ontap-san",
  "managementLIF": "10.0.0.1",
  "sanType": "fcp",
  "svm": "trident_svm",
  "username": "vsadmin",
  "password": "password",
  "useREST": true
}
```

Configuração do software Element

Além dos valores de configuração global, ao usar o software Element (NetApp HCI/SolidFire), essas opções estão disponíveis.

Opção	Descrição	Exemplo
`Endpoint`	/<login>:<password>/<mvip>/json-rpc/<element-version>	https://admin:admin@192.168.160.3/json-rpc/8.0
`SVIP`	Endereço IP iSCSI e porta	10,0.0,7:3260
`TenantName`	Locatário do SolidFireF para usar (criado se não for encontrado)	`docker`
`InitiatorIFace`	Especifique a interface ao restringir o tráfego iSCSI a uma interface não predefinida	`default`
`Types`	Especificações de QoS	Veja o exemplo abaixo
`LegacyNamePrefix`	Prefixo para instalações Trident atualizadas. Se você usou uma versão do Trident anterior a 1.3.2 e executar uma atualização com volumes existentes, precisará definir esse valor para acessar seus volumes antigos que foram mapeados pelo método de nome de volume.	`netappdvp-`

O `solidfire-san` driver não suporta Docker Swarm.

Exemplo de arquivo de configuração de software Element

```
{
  "version": 1,
  "storageDriverName": "solidfire-san",
  "Endpoint": "https://admin:admin@192.168.160.3/json-rpc/8.0",
```

```
    "SVIP": "10.0.0.7:3260",
    "TenantName": "docker",
    "InitiatorIFace": "default",
    "Types": [
      {
        "Type": "Bronze",
        "Qos": {
          "minIOPS": 1000,
          "maxIOPS": 2000,
          "burstIOPS": 4000
        }
      },
      {
        "Type": "Silver",
        "Qos": {
          "minIOPS": 4000,
          "maxIOPS": 6000,
          "burstIOPS": 8000
        }
      },
      {
        "Type": "Gold",
        "Qos": {
          "minIOPS": 6000,
          "maxIOPS": 8000,
          "burstIOPS": 10000
        }
      }
    ]
}
```

Problemas e limitações conhecidos

Encontre informações sobre problemas e limitações conhecidos ao usar o Trident com Docker.

A atualização do plug-in de volume do Docker do Trident para 20,10 e posterior a partir de versões mais antigas resulta em falha de atualização com o erro de nenhum arquivo ou diretório.

Solução alternativa

1. Desative o plugin.

```
docker plugin disable -f netapp:latest
```

2. Remova o plugin.

```
docker plugin rm -f netapp:latest
```

3. Reinstale o plugin fornecendo o parâmetro extra `config`.

```
docker plugin install netapp/trident-plugin:20.10 --alias netapp --grant
-all-permissions config=config.json
```

Os nomes dos volumes devem ter um mínimo de 2 carateres.

OBSERVAÇÃO	Esta é uma limitação de cliente Docker. O cliente interpretará um único nome de caractere como sendo um caminho do Windows. "Veja o bug 25773".

O Docker Swarm tem certos comportamentos que impedem o Trident de suportá-lo com cada combinação de armazenamento e driver.

- Docker Swarm atualmente faz uso do nome do volume em vez de ID do volume como seu identificador de volume exclusivo.
- As solicitações de volume são enviadas simultaneamente para cada nó em um cluster Swarm.
- Plugins de volume (incluindo Trident) devem ser executados independentemente em cada nó em um cluster Swarm. Devido à forma como o ONTAP funciona e como os `ontap-nas` drivers e `ontap-san` funcionam, eles são os únicos que podem operar dentro dessas limitações.

O resto dos pilotos estão sujeitos a problemas como condições de corrida que podem resultar na criação de um grande número de volumes para uma única solicitação sem um claro "vencedor"; por exemplo, o elemento tem um recurso que permite que os volumes tenham o mesmo nome, mas IDs diferentes.

O NetApp forneceu feedback à equipe do Docker, mas não tem qualquer indicação de recurso futuro.

Se um FlexGroup estiver sendo provisionado, o ONTAP não provisiona um segundo FlexGroup se o segundo FlexGroup tiver um ou mais agregados em comum com o FlexGroup sendo provisionado.

Práticas recomendadas e recomendações

Recomendações de implantação do Trident

Use as recomendações listadas aqui quando implantar o Trident.

Implante em um namespace dedicado

"Namespaces" fornecer separação administrativa entre diferentes aplicações e são uma barreira para o compartilhamento de recursos. Por exemplo, um PVC de um namespace não pode ser consumido de outro. O Trident fornece recursos PV para todos os namespaces no cluster do Kubernetes e, consequentemente, aproveita uma conta de serviço que elevou o Privileges.

Além disso, o acesso ao pod Trident pode permitir que um usuário acesse credenciais do sistema de storage e outras informações confidenciais. É importante garantir que os usuários de aplicativos e aplicativos de gerenciamento não tenham a capacidade de acessar as definições de objetos do Trident ou os próprios pods.

Use cotas e limites de intervalo para controlar o consumo de armazenamento

O Kubernetes tem dois recursos que, quando combinados, fornecem um mecanismo avançado para limitar o consumo de recursos pelas aplicações. O "mecanismo de cota de storage" permite que o administrador implemente limites de consumo globais e específicos de classe de storage, de contagem de objetos e capacidade em uma base por namespace. Além disso, o uso de a "limite de alcance" garante que as solicitações de PVC estejam dentro de um valor mínimo e máximo antes que a solicitação seja encaminhada para o provisionador.

Esses valores são definidos em uma base por namespace, o que significa que cada namespace deve ter valores definidos que se encaixam em seus requisitos de recursos. Consulte aqui para obter informações "como alavancar cotas" sobre .

Melhores práticas para a configuração de storage do Trident

Cada plataforma de storage do portfólio do NetApp tem funcionalidades exclusivas que beneficiam aplicações, em contêineres ou não.

Visão geral da plataforma

O Trident funciona com ONTAP e Element. Não há uma plataforma que seja mais adequada para todos os aplicativos e cenários do que outra, no entanto, as necessidades do aplicativo e da equipe que administra o dispositivo devem ser levadas em conta ao escolher uma plataforma.

Você deve seguir as práticas recomendadas de linha de base para o sistema operacional host com o protocolo que você está utilizando. Opcionalmente, você pode considerar a incorporação de práticas recomendadas de aplicativos, quando disponíveis, com configurações de backend, classe de armazenamento e PVC para otimizar o armazenamento para aplicativos específicos.

Práticas recomendadas de ONTAP e Cloud Volumes ONTAP

Conheça as práticas recomendadas para configurar o ONTAP e o Cloud Volumes ONTAP for Trident.

As recomendações a seguir são diretrizes para configuração do ONTAP para workloads em contêineres, que consomem volumes provisionados dinamicamente pelo Trident. Cada um deve ser considerado e avaliado quanto à adequação em seu ambiente.

Use SVM(s) dedicados ao Trident

As máquinas virtuais de storage (SVMs) fornecem isolamento e separação administrativa entre locatários em um sistema ONTAP. A dedicação de um SVM a aplicações permite a delegação do Privileges e permite aplicar práticas recomendadas para limitar o consumo de recursos.

Há várias opções disponíveis para o gerenciamento do SVM:

- Fornecer a interface de gerenciamento de cluster na configuração de back-end, juntamente com as credenciais apropriadas, e especificar o nome da SVM.
- Crie uma interface de gerenciamento dedicada ao SVM com o Gerenciador de sistemas do ONTAP ou a CLI.
- Compartilhe a função de gerenciamento com uma interface de dados NFS.

Em cada caso, a interface deve estar em DNS, e o nome DNS deve ser usado ao configurar o Trident. Isso ajuda a facilitar alguns cenários de DR, por exemplo, SVM-DR sem o uso de retenção de identidade de rede.

No entanto, não há preferência entre ter um LIF de gerenciamento dedicado ou compartilhado para o SVM, você deve garantir que suas políticas de segurança de rede estejam alinhadas com a abordagem escolhida. Independentemente disso, o LIF de gerenciamento deve ser acessível via DNS para facilitar a máxima flexibilidade deve "SVM-DR" ser usado em conjunto com o Trident.

Limite a contagem máxima de volume

Os sistemas de storage ONTAP têm uma contagem de volume máxima, que varia de acordo com a versão do software e a plataforma de hardware. "NetApp Hardware Universe"Consulte para obter informações sobre a sua plataforma específica e a versão do ONTAP para determinar os limites exatos. Quando a contagem de volume está esgotada, as operações de provisionamento falham não apenas para o Trident, mas para todas as solicitações de storage.

Os Trident `ontap-nas` e `ontap-san` os drivers provisionam um Flexvolume para cada volume persistente (PV) do Kubernetes criado. O `ontap-nas-economy` driver cria aproximadamente um Flexvolume para cada 200 PVS (configurável entre 50 e 300). O `ontap-san-economy` driver cria aproximadamente um Flexvolume para cada 100 PVS (configurável entre 50 e 200). Para evitar que o Trident consuma todos os volumes disponíveis no sistema de storage, defina um limite para o SVM. Você pode fazer isso a partir da linha de comando:

```
vserver modify -vserver <svm_name> -max-volumes <num_of_volumes>
```

O valor para `max-volumes` varia com base em vários critérios específicos para o seu ambiente:

- O número de volumes existentes no cluster do ONTAP
- O número de volumes que você espera provisionar fora do Trident para outras aplicações
- O número de volumes persistentes esperado para ser consumido pelas aplicações Kubernetes

O `max-volumes` valor é o total de volumes provisionados em todos os nós do cluster ONTAP e não em um nó ONTAP individual. Como resultado, você pode encontrar algumas condições em que um nó de cluster do

ONTAP pode ter muito mais ou menos volumes provisionados pelo Trident do que outro nó.

Por exemplo, um cluster de ONTAP de dois nós pode hospedar um máximo de 2000 volumes FlexVol. Ter a contagem de volume máxima definida para 1250 parece muito razoável. No entanto, se apenas "agregados" de um nó forem atribuídos à SVM, ou se os agregados atribuídos de um nó não puderem ser provisionados (por exemplo, devido à capacidade), o outro nó se tornará o destino de todos os volumes provisionados pelo Trident. Isso significa que o limite de volume pode ser alcançado para esse nó antes que o `max-volumes` valor seja atingido, resultando em impacto no Trident e em outras operações de volume que usam esse nó. **Você pode evitar essa situação garantindo que os agregados de cada nó no cluster sejam atribuídos ao SVM usado pelo Trident em números iguais.**

Clonar um volume

O NetApp Trident suporta a clonagem de volumes ao usar o `ontap-nas` , `ontap-san` , e `solidfire-san` drivers de armazenamento. Ao usar o `ontap-nas-flexgroup` ou `ontap-nas-economy` Drivers, clonagem não é suportada. Criar um novo volume a partir de um volume existente resultará na criação de um novo snapshot.

AVISO	Evite clonar um PVC associado a uma StorageClass diferente. Execute operações de clonagem dentro da mesma StorageClass para garantir a compatibilidade e evitar comportamentos inesperados.

Limite o tamanho máximo de volumes criados pelo Trident

Para configurar o tamanho máximo para volumes que podem ser criados pelo Trident, use o `limitVolumeSize` parâmetro em `backend.json` sua definição.

Além de controlar o tamanho do volume no storage array, você também deve utilizar os recursos do Kubernetes.

Limite o tamanho máximo de FlexVols criados pelo Trident

Para configurar o tamanho máximo para FlexVols usados como pools para drivers ONTAP-san-Economy e ONTAP-nas-Economy, use o `limitVolumePoolSize` parâmetro em sua `backend.json` definição.

Configure o Trident para usar o CHAP bidirecional

Você pode especificar o iniciador CHAP e os nomes de usuário e senhas de destino em sua definição de back-end e ter o Trident Enable CHAP no SVM. Usando o `useCHAP` parâmetro em sua configuração de back-end, o Trident autentica conexões iSCSI para backends ONTAP com CHAP.

Criar e usar uma política de QoS SVM

A utilização de uma política de QoS ONTAP aplicada à SVM limita o número de consumíveis de IOPS pelos volumes provisionados pelo Trident. Isso ajuda "evite um bully" a evitar que o volume fora de controle afete workloads fora do SVM do Trident.

Você pode criar uma política de QoS para o SVM em algumas etapas. Consulte a documentação da sua versão do ONTAP para obter as informações mais precisas. O exemplo abaixo cria uma política de QoS que limita o total de IOPS disponível para o SVM a 5000.

```
# create the policy group for the SVM
```

```
qos policy-group create -policy-group <policy_name> -vserver <svm_name>
-max-throughput 5000iops

# assign the policy group to the SVM, note this will not work
# if volumes or files in the SVM have existing QoS policies
vserver modify -vserver <svm_name> -qos-policy-group <policy_name>
```

Além disso, se a sua versão do ONTAP for compatível com ela, considere o uso de um mínimo de QoS para garantir uma taxa de transferência para workloads em contêineres. A QoS adaptável não é compatível com uma política de nível SVM.

O número de IOPS dedicado aos workloads em contêineres depende de muitos aspectos. Entre outras coisas, estas incluem:

- Outros workloads que usam o storage array. Se houver outras cargas de trabalho, não relacionadas à implantação do Kubernetes, utilizando os recursos de storage, deve-se tomar cuidado para garantir que essas cargas de trabalho não sejam acidentalmente afetadas.
- Workloads esperados em execução em contêineres. Se os workloads com requisitos de IOPS altos forem executados em contêineres, uma política de QoS baixa resulta em uma experiência ruim.

É importante lembrar que uma política de QoS atribuída no nível SVM resulta em todos os volumes provisionados ao SVM que compartilham o mesmo pool de IOPS. Se uma, ou um número pequeno, das aplicações em contêiner tiverem um requisito de IOPS alto, isso pode se tornar um bully para os outros workloads em contêiner. Se esse for o caso, você pode considerar o uso de automação externa para atribuir políticas de QoS por volume.

IMPORTANTE	Você deve atribuir o grupo de políticas de QoS ao SVM **somente** se a versão do ONTAP for anterior a 9,8.

Criar grupos de política de QoS para Trident

A qualidade do serviço (QoS) garante que a performance de workloads essenciais não é degradada pelos workloads da concorrência. Os grupos de política de QoS do ONTAP fornecem opções de QoS para volumes e permitem que os usuários definam o limite máximo de taxa de transferência para um ou mais workloads. Para obter mais informações sobre QoS, "Garantir taxa de transferência com QoS" consulte . É possível especificar grupos de políticas de QoS no back-end ou em um pool de storage, e eles são aplicados a cada volume criado nesse pool ou back-end.

O ONTAP tem dois tipos de grupos de política de QoS: Tradicional e adaptável. Os grupos de políticas tradicionais fornecem uma taxa de transferência máxima fixa (ou mínima, em versões posteriores) em IOPS. O serviço adaptável dimensiona automaticamente a taxa de transferência para o tamanho do workload, mantendo a taxa de IOPS para TBs|GBs conforme o tamanho do workload muda. Isso proporciona uma vantagem significativa ao gerenciar centenas ou milhares de workloads em uma implantação grande.

Considere o seguinte ao criar grupos de política de QoS:

- Você deve definir a `qosPolicy` chave no `defaults` bloco da configuração de back-end. Veja o seguinte exemplo de configuração de back-end:

```
---
version: 1
```

```
storageDriverName: ontap-nas
managementLIF: 0.0.0.0
dataLIF: 0.0.0.0
svm: svm0
username: user
password: pass
defaults:
  qosPolicy: standard-pg
storage:
  - labels:
      performance: extreme
    defaults:
      adaptiveQosPolicy: extremely-adaptive-pg
  - labels:
      performance: premium
    defaults:
      qosPolicy: premium-pg
```

- Você deve aplicar os grupos de políticas por volume, para que cada volume obtenha toda a taxa de transferência, conforme especificado pelo grupo de políticas. Grupos de políticas compartilhadas não são suportados.

Para obter mais informações sobre grupos de políticas de QoS, "Referência do comando ONTAP" consulte .

Limitar o acesso a recursos de storage aos membros do cluster do Kubernetes

Limitar o acesso a volumes NFS, iSCSI LUNs e FC LUNs criados pelo Trident é um componente essencial da postura de segurança para a implantação do Kubernetes. Isso impede que os hosts que não fazem parte do cluster do Kubernetes acessem os volumes e potencialmente modifiquem os dados inesperadamente.

É importante entender que os namespaces são o limite lógico dos recursos no Kubernetes. A suposição é que os recursos no mesmo namespace são capazes de ser compartilhados, no entanto, é importante, não há capacidade entre namespace. Isso significa que, embora os PVS sejam objetos globais, quando vinculados a um PVC, eles só são acessíveis por pods que estão no mesmo namespace. **É fundamental garantir que os namespaces sejam usados para fornecer separação quando apropriado.**

A principal preocupação da maioria das organizações com relação à segurança de dados em um contexto do Kubernetes é que um processo em um contêiner pode acessar o storage montado no host, mas que não se destina ao contêiner. "Namespaces" foram concebidos para evitar este tipo de compromisso. No entanto, há uma exceção: Contentores privilegiados.

Um contentor privilegiado é aquele que é executado com permissões substancialmente mais no nível do host do que o normal. Estes não são negados por padrão, portanto, certifique-se de desativar a capacidade "diretivas de segurança do pod" usando o .

Para volumes em que o acesso é desejado tanto do Kubernetes quanto de hosts externos, o storage deve ser gerenciado de maneira tradicional, com o PV introduzido pelo administrador e não gerenciado pelo Trident. Isso garante que o volume de storage seja destruído somente quando o Kubernetes e os hosts externos forem desconetados e não estiverem mais usando o volume. Além disso, é possível aplicar uma política de exportação personalizada, que permite o acesso dos nós de cluster do Kubernetes e dos servidores direcionados fora do cluster do Kubernetes.

Para implantações com nós de infraestrutura dedicados (por exemplo, OpenShift) ou outros nós que não conseguem programar aplicativos de usuário, políticas de exportação separadas devem ser usadas para limitar ainda mais o acesso aos recursos de storage. Isso inclui a criação de uma política de exportação para serviços que são implantados nesses nós de infraestrutura (por exemplo, os serviços de métricas e Registro OpenShift) e aplicativos padrão que são implantados em nós que não são de infraestrutura.

Use uma política de exportação dedicada

Você deve garantir que existe uma política de exportação para cada back-end que permita somente o acesso aos nós presentes no cluster do Kubernetes. O Trident pode criar e gerenciar automaticamente políticas de exportação. Dessa forma, o Trident limita o acesso aos volumes provisionados por TI aos nós no cluster do Kubernetes e simplifica a adição/exclusão de nós.

Como alternativa, você também pode criar uma política de exportação manualmente e preenchê-la com uma ou mais regras de exportação que processam cada solicitação de acesso de nó:

- Use o `vserver export-policy create` comando ONTAP CLI para criar a política de exportação.
- Adicione regras à política de exportação usando o `vserver export-policy rule create` comando ONTAP CLI.

Executar esses comandos permite restringir quais nós do Kubernetes têm acesso aos dados.

`showmount`Desativar o SVM da aplicação

O `showmount` recurso permite que um cliente NFS consulte o SVM para obter uma lista de exportações de NFS disponíveis. Um pod implantado no cluster do Kubernetes pode emitir o `showmount -e` comando no e receber uma lista de montagens disponíveis, incluindo aquelas às quais ele não tem acesso. Embora isso, por si só, não seja um compromisso de segurança, ele fornece informações desnecessárias potencialmente ajudando um usuário não autorizado a se conetar a uma exportação NFS.

Você deve desativar `showmount` usando o comando ONTAP CLI no nível da SVM:

```
vserver nfs modify -vserver <svm_name> -showmount disabled
```

Práticas recomendadas da SolidFire

Conheça as práticas recomendadas para configurar o armazenamento SolidFire para Trident.

Crie uma conta SolidFire

Cada conta do SolidFire representa um proprietário de volume exclusivo e recebe seu próprio conjunto de credenciais do Protocolo de Autenticação de desafio-aperto (CHAP). Você pode acessar volumes atribuídos a uma conta usando o nome da conta e as credenciais CHAP relativas ou por meio de um grupo de acesso de volume. Uma conta pode ter até dois mil volumes atribuídos a ela, mas um volume pode pertencer a apenas uma conta.

Crie uma política de QoS

Use as políticas de qualidade do serviço (QoS) do SolidFire se quiser criar e salvar uma configuração padronizada de qualidade do serviço que pode ser aplicada a muitos volumes.

Você pode definir parâmetros de QoS em uma base por volume. O desempenho de cada volume pode ser

garantido definindo três parâmetros configuráveis que definem a QoS: Min IOPS, Max IOPS e Burst IOPS.

Aqui estão os possíveis valores de IOPS mínimo, máximo e de pico sazonal para o tamanho de bloco 4Kb.

Parâmetro IOPS	Definição	Valor mín	Valor padrão	Valor máximo (4Kb)
IOPS mín	O nível garantido de desempenho para um volume.	50	50	15000
IOPS máx	O desempenho não excederá este limite.	50	15000	200.000
IOPS de explosão	Máximo de IOPS permitido em um cenário de pico curto.	50	15000	200.000

OBSERVAÇÃO	Embora o IOPS máximo e o IOPS Burst possam ser definidos até 200.000 K, o desempenho máximo real de um volume é limitado pelo uso do cluster e pelo desempenho por nó.

O tamanho do bloco e a largura de banda têm influência direta sobre o número de IOPS. À medida que os tamanhos dos blocos aumentam, o sistema aumenta a largura de banda para um nível necessário para processar os blocos maiores. À medida que a largura de banda aumenta, o número de IOPS que o sistema é capaz de atingir diminui. Consulte "SolidFire qualidade do serviço" para mais informações sobre QoS e desempenho.

Autenticação SolidFire

O Element suporta dois métodos de autenticação: CHAP e volume Access Groups (VAG). O CHAP usa o protocolo CHAP para autenticar o host no back-end. Os grupos de acesso de volume controlam o acesso aos volumes que ele provisiona. O NetApp recomenda usar o CHAP para autenticação, pois é mais simples e não tem limites de escala.

OBSERVAÇÃO	O Trident com o provisionador de CSI aprimorado suporta o uso da autenticação CHAP. Os VAG só devem ser utilizados no modo de funcionamento tradicional não CSI.

A autenticação CHAP (verificação de que o iniciador é o usuário de volume pretendido) é suportada apenas com controle de acesso baseado em conta. Se você estiver usando CHAP para autenticação, duas opções estão disponíveis: CHAP unidirecional e CHAP bidirecional. O CHAP unidirecional autentica o acesso ao volume usando o nome da conta do SolidFire e o segredo do iniciador. A opção CHAP bidirecional fornece a maneira mais segura de autenticar o volume porque o volume autentica o host através do nome da conta e do segredo do iniciador e, em seguida, o host autentica o volume através do nome da conta e do segredo de destino.

No entanto, se o CHAP não puder ser ativado e os VAG forem necessários, crie o grupo de acesso e adicione os iniciadores e volumes do host ao grupo de acesso. Cada IQN que você adicionar a um grupo de acesso pode acessar cada volume no grupo com ou sem autenticação CHAP. Se o iniciador iSCSI estiver configurado para usar autenticação CHAP, o controle de acesso baseado em conta será usado. Se o iniciador iSCSI não estiver configurado para usar a autenticação CHAP, o controle de acesso ao grupo de acesso de volume será usado.

Onde encontrar mais informações?

Alguns dos documentos de melhores práticas estão listados abaixo. PESQUISE na "Biblioteca NetApp" para as versões mais atuais.

ONTAP

- "Guia de práticas recomendadas e implementação de NFS"
- "Administração de SAN" (Para iSCSI)
- "Configuração iSCSI Express para RHEL"

Software Element

- "Configurando o SolidFire para Linux"

NetApp HCI

- "Pré-requisitos de implantação do NetApp HCI"
- "Acesse o mecanismo de implantação do NetApp"

Informações sobre as melhores práticas de aplicação

- "Melhores práticas para MySQL no ONTAP"
- "Melhores práticas para MySQL no SolidFire"
- "NetApp SolidFire e Cassandra"
- "Práticas recomendadas da Oracle no SolidFire"
- "Melhores práticas do PostgreSQL no SolidFire"

Nem todos os aplicativos têm diretrizes específicas, é importante trabalhar com sua equipe do NetApp e usar o "Biblioteca NetApp" para encontrar a documentação mais atualizada.

Integre o Trident com o Kubernetes

Para integrar o Trident, os seguintes elementos de design e arquitetura exigem integração: Seleção e implantação de drivers, design de classe de storage, design de pool virtual, impactos da reivindicação de volume persistente (PVC) no provisionamento de storage, operações de volume e implantação de serviços OpenShift usando o Trident.

Seleção e implantação do driver

Selecione e implante um driver de back-end para seu sistema de storage.

Drivers de back-end do ONTAP

Os drivers de back-end do ONTAP são diferenciados pelo protocolo usado e pelo modo como os volumes são provisionados no sistema de storage. Portanto, tenha cuidado ao decidir qual driver implantar.

Em um nível mais alto, se seu aplicativo tiver componentes que precisam de armazenamento compartilhado (vários pods acessando o mesmo PVC), os drivers baseados em nas seriam a escolha padrão, enquanto os drivers iSCSI baseados em bloco atendem às necessidades de armazenamento não compartilhado. Escolha o

protocolo com base nos requisitos da aplicação e no nível de conforto das equipes de armazenamento e infraestrutura. De um modo geral, há pouca diferença entre eles para a maioria dos aplicativos, portanto, muitas vezes a decisão é baseada na necessidade ou não de armazenamento compartilhado (onde mais de um pod precisará de acesso simultâneo).

Os drivers de back-end ONTAP disponíveis são:

- `ontap-nas`: Cada PV provisionado é um Flexvolume ONTAP completo.
- `ontap-nas-economy`: Cada PV provisionado é uma qtree, com um número configurável de qtrees por Flexvolume (o padrão é 200).
- `ontap-nas-flexgroup`: Cada PV provisionado como um ONTAP FlexGroup completo e todos os agregados atribuídos a um SVM são usados.
- `ontap-san`: Cada PV provisionado é um LUN dentro de seu próprio Flexvolume.
- `ontap-san-economy`: Cada PV provisionado é um LUN, com um número configurável de LUNs por Flexvolume (o padrão é 100).

A escolha entre os três drivers nas tem algumas ramificações para os recursos, que são disponibilizados para o aplicativo.

Observe que, nas tabelas abaixo, nem todos os recursos são expostos por meio do Trident. Alguns devem ser aplicados pelo administrador de armazenamento após o provisionamento, se essa funcionalidade for desejada. As notas de rodapé sobrescritas distinguem a funcionalidade por recurso e driver.

Drivers nas ONTAP	Instantâneos	Clones	Políticas de exportação dinâmicas	Ligação múltipla	QoS	Redimensionar	Replicação
`ontap-nas`	Sim	Sim	Nota de rodapé:5[]	Sim	Nota de rodapé:1[]	Sim	Nota de rodapé:1[]
`ontap-nas-economy`	Nota de rodapé:3[]	Nota de rodapé:3[]	Nota de rodapé:5[]	Sim	Nota de rodapé:3[]	Sim	Nota de rodapé:3[]
`ontap-nas-flexgroup`	Nota de rodapé:1[]	NÃO	Nota de rodapé:5[]	Sim	Nota de rodapé:1[]	Sim	Nota de rodapé:1[]

A Trident oferece 2 drivers SAN para ONTAP, cujos recursos são mostrados abaixo.

Controladores SAN ONTAP	Instantâneos	Clones	Ligação múltipla	CHAP bidirecional	QoS	Redimensionar	Replicação
`ontap-san`	Sim	Sim	Nota de rodapé:4[]	Sim	Nota de rodapé:1[]	Sim	Nota de rodapé:1[]
`ontap-san-economy`	Sim	Sim	Nota de rodapé:4[]	Sim	Nota de rodapé:3[]	Sim	Nota de rodapé:3[]

> Nota de rodapé para as tabelas acima: Yes[1]: Não gerenciado por Trident Yes[2]: Gerenciado por Trident, mas não PV granular NO[3]: Não gerenciado por Trident e

> não PV granular Yes[4]: Suportado para volumes de bloco bruto Yes[5]: Suportado por Trident

Os recursos que não são granulares PV são aplicados a todo o Flexvolume e todos os PVS (ou seja, qtrees ou LUNs em FlexVols compartilhados) compartilharão um cronograma comum.

Como podemos ver nas tabelas acima, grande parte da funcionalidade entre `ontap-nas` o e `ontap-nas-economy` é a mesma. No entanto, como o `ontap-nas-economy` motorista limita a capacidade de controlar o cronograma em granularidade por PV, isso pode afetar sua recuperação de desastres e Planejamento de backup em particular. Para as equipes de desenvolvimento que desejam utilizar a funcionalidade de clone de PVC no storage ONTAP, isso só é possível ao usar os `ontap-nas` drivers , `ontap-san` ou `ontap-san-economy` .

OBSERVAÇÃO	O `solidfire-san` driver também é capaz de clonar PVCs.

Drivers de back-end do Cloud Volumes ONTAP

O Cloud Volumes ONTAP fornece controle de dados junto a recursos de storage de classe empresarial para vários casos de uso, incluindo compartilhamentos de arquivos e storage em nível de bloco, atendendo aos protocolos nas e SAN (NFS, SMB/CIFS e iSCSI). Os drivers compatíveis para o Cloud volume ONTAP são `ontap-nas`, `ontap-nas-economy` `ontap-san` e `ontap-san-economy`. Eles são aplicáveis ao Cloud volume ONTAP para Azure, Cloud volume ONTAP para GCP.

Drivers de back-end do Amazon FSX para ONTAP

O Amazon FSX for NetApp ONTAP permite que você aproveite os recursos, o desempenho e os recursos administrativos do NetApp que você já conhece, ao mesmo tempo em que aproveita a simplicidade, a agilidade, a segurança e a escalabilidade do armazenamento de dados na AWS. O FSX para ONTAP suporta muitos recursos do sistema de arquivos ONTAP e APIs de administração. Os drivers compatíveis para o Cloud volume ONTAP são `ontap-nas`, `ontap-nas-economy`, `ontap-nas-flexgroup` `ontap-san` e `ontap-san-economy`.

Drivers de back-end NetApp HCI/SolidFire

O `solidfire-san` driver usado com as plataformas NetApp HCI/SolidFire ajuda o administrador a configurar um back-end Element para Trident com base nos limites de QoS. Se você quiser projetar seu back-end para definir os limites de QoS específicos nos volumes provisionados pelo Trident, use o `type` parâmetro no arquivo de back-end. O administrador também pode restringir o tamanho do volume que pode ser criado no armazenamento usando o `limitVolumeSize` parâmetro. Atualmente, recursos de armazenamento de elementos, como redimensionamento de volume e replicação de volume, não são suportados pelo `solidfire-san` driver. Essas operações devem ser feitas manualmente por meio da IU da Web do Element Software.

Controlador SolidFire	Instantâneos	Clones	Ligação múltipla	CHAP	QoS	Redimensionar	Replicação
`solidfire-san`	Sim	Sim	Nota de rodapé:2[]	Sim	Sim	Sim	Nota de rodapé:1[]

> Nota de rodapé: Yes[1]: Não gerenciado por Trident Yes[2]: Suportado para volumes de blocos brutos

Drivers de back-end do Azure NetApp Files

Trident usa o `azure-netapp-files` driver para gerenciar o serviço "Azure NetApp Files".

Mais informações sobre esse driver e como configurá-lo podem ser encontradas no "Configuração de back-end do Trident para Azure NetApp Files".

Controlador Azure NetApp Files	Instantâneos	Clones	Ligação múltipla	QoS	Expandir	Replicação
`azure-netapp-files`	Sim	Sim	Sim	Sim	Sim	Nota de rodapé:1[]

> Nota de rodapé: Yes[1]: Não gerenciado por Trident

Drivers de back-end do Google Cloud NetApp Volumes

O Trident oferece suporte ao `google-cloud-netapp-volumes` driver para cargas de trabalho NAS e ao `google-cloud-netapp-volumes-san` para cargas de trabalho SAN (iSCSI). Em clusters do OpenShift executados no GCP, ambos os drivers oferecem suporte à Google Cloud Workload Identity Federation a partir do Trident 26.10. Isso inclui a OpenShift Container Platform autogerenciada (OCP, incluindo implantações IPI) e o OpenShift Dedicated (OSD).

Para a configuração de backend e os pré-requisitos do WIF, consulte "Configurar Google Cloud NetApp Volumes" e "Configurar o Google Cloud NetApp Volumes para cargas de trabalho SAN".

Design da classe de armazenamento

As classes de armazenamento individuais precisam ser configuradas e aplicadas para criar um objeto Classe de armazenamento Kubernetes. Esta seção discute como projetar uma classe de armazenamento para seu aplicativo.

Utilização específica no back-end

A filtragem pode ser usada dentro de um objeto de classe de armazenamento específico para determinar qual pool de armazenamento ou conjunto de pools devem ser usados com essa classe de armazenamento específica. Três conjuntos de filtros podem ser definidos na Classe de armazenamento: `storagePools`, `additionalStoragePools` E/ou `excludeStoragePools`.

O `storagePools` parâmetro ajuda a restringir o armazenamento ao conjunto de pools que correspondem a quaisquer atributos especificados. O `additionalStoragePools` parâmetro é usado para estender o conjunto de pools que o Trident usa para provisionar junto com o conjunto de pools selecionados pelos atributos e `storagePools` parâmetros. Você pode usar um parâmetro sozinho ou ambos juntos para garantir que o conjunto apropriado de pools de armazenamento esteja selecionado.

O `excludeStoragePools` parâmetro é usado para excluir especificamente o conjunto listado de pools que correspondem aos atributos.

Emular políticas de QoS

Se você quiser criar classes de armazenamento para emular políticas de qualidade de Serviço, crie uma Classe de armazenamento com o `media` atributo como `hdd` ou `ssd`. Com base no `media` atributo mencionado na classe de storage, o Trident selecionará o back-end apropriado que serve `hdd` ou `ssd` agrega

para corresponder ao atributo de Mídia e direcionará o provisionamento dos volumes para o agregado específico. Portanto, podemos criar uma classe de armazenamento PREMIUM que teria um conjunto de atributos, `ssd` que `media` poderia ser classificado como a política de QoS PREMIUM. Podemos criar outro PADRÃO de classe de armazenamento que teria o atributo de Mídia definido como "hdd", que poderia ser classificado como a política de QoS PADRÃO. Também podemos usar o atributo "IOPS" na classe de armazenamento para redirecionar o provisionamento para um dispositivo Element que pode ser definido como uma Política de QoS.

Utilize o back-end com base em recursos específicos

As classes de storage podem ser projetadas para direcionar o provisionamento de volume em um back-end específico, no qual recursos como provisionamento fino e espesso, snapshots, clones e criptografia são ativados. Para especificar qual armazenamento usar, crie classes de armazenamento que especifiquem o back-end apropriado com o recurso necessário habilitado.

Pools virtuais

Os pools virtuais estão disponíveis para todos os backends do Trident. Você pode definir pools virtuais para qualquer back-end, usando qualquer driver fornecido pelo Trident.

Os pools virtuais permitem que um administrador crie um nível de abstração sobre backends que pode ser referenciado por meio de classes de armazenamento, para maior flexibilidade e colocação eficiente de volumes em backends. Diferentes backends podem ser definidos com a mesma classe de serviço. Além disso, vários pools de storage podem ser criados no mesmo back-end, mas com características diferentes. Quando uma Classe de armazenamento é configurada com um seletor com as etiquetas específicas, o Trident escolhe um backend que corresponde a todas as etiquetas do seletor para colocar o volume. Se as etiquetas do seletor de Classe de armazenamento corresponder a vários pools de armazenamento, o Trident escolherá um deles para provisionar o volume.

Design de pool virtual

Ao criar um backend, geralmente é possível especificar um conjunto de parâmetros. Era impossível para o administrador criar outro backend com as mesmas credenciais de armazenamento e um conjunto diferente de parâmetros. Com a introdução de pools virtuais, esse problema foi amenizado. Um pool virtual é uma abstração de nível introduzida entre o backend e a Classe de Armazenamento do Kubernetes para que o administrador possa definir parâmetros junto com rótulos que podem ser referenciados por meio das Classes de Armazenamento do Kubernetes como um seletor, de forma independente do backend. Pools virtuais podem ser definidos para todos os backends NetApp suportados com o Trident. Essa lista inclui SolidFire/ NetApp HCI, ONTAP, bem como Azure NetApp Files.

OBSERVAÇÃO	Ao definir pools virtuais, é recomendável não tentar reorganizar a ordem dos pools virtuais existentes em uma definição de back-end. Também é aconselhável não editar/modificar atributos para um pool virtual existente e definir um novo pool virtual.

Emulando diferentes níveis de serviço/QoS

É possível projetar pools virtuais para emular classes de serviço. Usando a implementação do pool virtual para o Cloud volume Service for Azure NetApp Files, vamos examinar como podemos configurar diferentes classes de serviço. Configure o back-end do Azure NetApp Files com vários rótulos, representando diferentes níveis de desempenho. Defina `servicelevel` Aspect para o nível de desempenho apropriado e adicione outros aspetos necessários em cada rótulo. Agora crie diferentes classes de armazenamento do Kubernetes que mapeariam para diferentes pools virtuais. Usando o `parameters.selector` campo, cada StorageClass

chama quais pools virtuais podem ser usados para hospedar um volume.

Atribuir um conjunto específico de aspetos

Vários pools virtuais com um conjunto específico de aspectos podem ser projetados a partir de um único back-end de storage. Para fazer isso, configure o back-end com vários rótulos e defina os aspetos necessários em cada rótulo. Agora crie diferentes classes de armazenamento do Kubernetes usando o `parameters.selector` campo que mapearia para diferentes pools virtuais. Os volumes que são provisionados no back-end terão os aspetos definidos no pool virtual escolhido.

Caraterísticas de PVC que afetam o provisionamento de armazenamento

Alguns parâmetros além da classe de armazenamento solicitada podem afetar o processo de decisão de provisionamento do Trident ao criar um PVC.

Modo de acesso

Ao solicitar armazenamento através de um PVC, um dos campos obrigatórios é o modo de acesso. O modo desejado pode afetar o back-end selecionado para hospedar a solicitação de armazenamento.

O Trident tentará corresponder ao protocolo de armazenamento utilizado com o método de acesso especificado de acordo com a seguinte matriz. Isso é independente da plataforma de storage subjacente.

	ReadWriteOnce	ReadOnlyMany	ReadWriteMany
ISCSI	Sim	Sim	Sim (bloco bruto)
NFS	Sim	Sim	Sim

Uma solicitação de um PVC ReadWriteMany enviado para uma implantação do Trident sem um back-end NFS configurado resultará em nenhum volume sendo provisionado. Por esse motivo, o solicitante deve usar o modo de acesso apropriado para sua aplicação.

Operações de volume

Modificar volumes persistentes

Volumes persistentes são, com duas exceções, objetos imutáveis no Kubernetes. Uma vez criados, a política de recuperação e o tamanho podem ser modificados. No entanto, isso não impede que alguns aspetos do volume sejam modificados fora do Kubernetes. Isso pode ser desejável para personalizar o volume para aplicações específicas, para garantir que a capacidade não seja consumida acidentalmente ou simplesmente mover o volume para um controlador de armazenamento diferente por qualquer motivo.

OBSERVAÇÃO	No momento, os provisionadores in-tree do Kubernetes não são compatíveis com operações de redimensionamento de volume para PVS NFS, iSCSI ou FC. O Trident é compatível com a expansão de volumes NFS, iSCSI e FC.

Os detalhes de ligação do PV não podem ser modificados após a criação.

Criar snapshots de volume sob demanda

O Trident é compatível com a criação de snapshot de volume sob demanda e a criação de PVCs a partir de snapshots usando a estrutura CSI. Os snapshots fornecem um método conveniente de manter cópias pontuais dos dados e têm um ciclo de vida independente do PV de origem no Kubernetes. Esses snapshots

podem ser usados para clonar PVCs.

Criar volumes a partir de instantâneos

O Trident também suporta a criação de PersistentVolumes a partir de instantâneos de volume. Para conseguir isso, basta criar um PersistentVolumeClaim e mencionar o `datasource` como o instantâneo necessário a partir do qual o volume precisa ser criado. O Trident manipulará esse PVC criando um volume com os dados presentes no instantâneo. Com esse recurso, é possível duplicar dados entre regiões, criar ambientes de teste, substituir um volume de produção danificado ou corrompido em sua totalidade, ou recuperar arquivos e diretórios específicos e transferi-los para outro volume anexado.

Mover volumes no cluster

Os administradores de storage podem mover volumes entre agregados e controladores no cluster ONTAP sem interrupções para o consumidor de storage. Essa operação não afeta o Trident ou o cluster do Kubernetes, contanto que o agregado de destino seja aquele ao qual o SVM que o Trident está usando tenha acesso. Importante, se o agregado tiver sido adicionado recentemente ao SVM, o back-end precisará ser atualizado readicionando-o ao Trident. Isso fará com que o Trident faça o inventário novamente da SVM para que o novo agregado seja reconhecido.

No entanto, mover volumes entre backends não é suportado automaticamente pelo Trident. Isso inclui entre SVMs no mesmo cluster, entre clusters ou em uma plataforma de storage diferente (mesmo que esse sistema de storage seja conetado ao Trident).

Se um volume for copiado para outro local, o recurso de importação de volume pode ser usado para importar volumes atuais para o Trident.

Expanda volumes

O Trident suporta o redimensionamento de PVs NFS, iSCSI e FC. Isso permite que os usuários redimensionem seus volumes diretamente através da camada Kubernetes. A expansão de volume é possível para todas as principais plataformas de armazenamento NetApp , incluindo ONTAP e back-ends SolidFire/ NetApp HCI . Para permitir uma possível expansão posterior, defina `allowVolumeExpansion` para `true` na sua StorageClass associada ao volume. Sempre que o Volume Persistente precisar ser redimensionado, edite o `spec.resources.requests.storage` anotação na Solicitação de Volume Persistente referente ao tamanho de volume necessário. O Trident se encarregará automaticamente de redimensionar o volume no cluster de armazenamento.

Importar um volume existente para o Kubernetes

A importação de volumes permite importar um volume de armazenamento existente para um ambiente Kubernetes. Isso é atualmente suportado pelo `ontap-nas` , `ontap-nas-flexgroup` , `solidfire-san` , e `azure-netapp-files` motoristas. Essa funcionalidade é útil ao migrar um aplicativo existente para o Kubernetes ou em cenários de recuperação de desastres.

Ao usar o ONTAP e `solidfire-san` os drivers, use o comando `tridentctl import volume <backend-name> <volume-name> -f /path/pvc.yaml` para importar um volume existente para o Kubernetes para ser gerenciado pelo Trident. O arquivo de PVC YAML ou JSON usado no comando volume de importação aponta para uma classe de storage que identifica o Trident como o provisionador. Ao usar um back-end NetApp HCI/SolidFire, certifique-se de que os nomes de volume sejam exclusivos. Se os nomes de volume forem duplicados, clone o volume para um nome exclusivo para que o recurso de importação de volume possa distinguir entre eles.

Se o `azure-netapp-files` O driver está sendo usado, use o comando `tridentctl import volume`

`<backend-name> <volume path> -f /path/pvc.yaml` Importar o volume para o Kubernetes para ser gerenciado pelo Trident. Isso garante uma referência de volume única.

Quando o comando acima é executado, o Trident irá encontrar o volume no backend e ler o seu tamanho. Ele irá adicionar automaticamente (e substituir, se necessário) o tamanho de volume do PVC configurado. O Trident então cria o novo PV e o Kubernetes liga o PVC ao PV.

Se um recipiente fosse implantado de modo que fosse necessário o PVC importado específico, ele permaneceria em um estado pendente até que o par PVC/PV seja vinculado através do processo de importação de volume. Depois que o par de PVC / PV são ligados, o recipiente deve surgir, desde que não haja outros problemas.

Serviço de registo

A implantação e o gerenciamento do armazenamento para o Registro foram documentados "NetApp.io" "blog"no .

Serviço de registo

Assim como outros serviços OpenShift, o serviço de log é implantado usando o Ansible com parâmetros de configuração fornecidos pelo arquivo de inventário, também conhecido como hosts, fornecidos ao manual de estratégia. Há dois métodos de instalação que serão abordados: Implantação de logs durante a instalação inicial do OpenShift e implantação de logs após a instalação do OpenShift.

CUIDADO	A partir do Red Hat OpenShift versão 3,9, a documentação oficial recomenda contra o NFS para o serviço de log devido a preocupações com a corrupção de dados. Isso é baseado no teste da Red Hat de seus produtos. O servidor NFS do ONTAP não tem esses problemas e pode facilmente fazer backup de uma implantação de log. Em última análise, a escolha do protocolo para o serviço de Registro é sua, apenas saiba que ambos funcionarão muito bem ao usar plataformas NetApp e não há motivo para evitar o NFS se essa for sua preferência.

Se você optar por usar o NFS com o serviço de log, precisará definir a variável Ansible `openshift_enable_unsupported_configurations` para `true` evitar que o instalador falhe.

Comece agora

O serviço de log pode, opcionalmente, ser implantado tanto para aplicativos quanto para as operações principais do próprio cluster OpenShift. Se você optar por implantar o Registro de operações, especificando a variável `openshift_logging_use_ops` como `true`, duas instâncias do serviço serão criadas. As variáveis que controlam a instância de log para operações contêm "OPS" nelas, enquanto a instância para aplicativos não.

A configuração das variáveis do Ansible de acordo com o método de implantação é importante para garantir que o storage correto seja utilizado pelos serviços subjacentes. Vejamos as opções para cada um dos métodos de implantação.

OBSERVAÇÃO	As tabelas abaixo contêm apenas as variáveis relevantes para configuração de storage conforme relacionado ao serviço de registro de logs. Você pode encontrar outras opções em "Documentação de Registro do Red Hat OpenShift" que devem ser revisadas, configuradas e utilizadas de acordo com a sua implementação.

As variáveis na tabela abaixo resultarão no manual do Ansible criando um PV e PVC para o serviço de

Registro usando os detalhes fornecidos. Esse método é significativamente menos flexível do que usar o manual de instalação de componentes após a instalação do OpenShift, no entanto, se você tiver volumes existentes disponíveis, é uma opção.

Variável	Detalhes
`openshift_logging_storage_kind`	Defina como `nfs` para que o instalador crie um NFS PV para o serviço de log.
`openshift_logging_storage_host`	O nome do host ou endereço IP do host NFS. Isso deve ser definido como dataLIF para sua máquina virtual.
`openshift_logging_storage_nfs_directory`	O caminho de montagem para a exportação NFS. Por exemplo, se o volume for juntado como `/openshift_logging`, você usaria esse caminho para essa variável.
`openshift_logging_storage_volume_name`	O nome, por exemplo `pv_ose_logs`, do PV a criar.
`openshift_logging_storage_volume_size`	O tamanho da exportação NFS, por `100Gi` exemplo .

Se o cluster do OpenShift já estiver em execução e, portanto, o Trident tiver sido implantado e configurado, o instalador poderá usar o provisionamento dinâmico para criar os volumes. As variáveis a seguir precisarão ser configuradas.

Variável	Detalhes
`openshift_logging_es_pvc_dynamic`	Defina como verdadeiro para usar volumes provisionados dinamicamente.
`openshift_logging_es_pvc_storage_class_name`	O nome da classe de armazenamento que será usado no PVC.
`openshift_logging_es_pvc_size`	O tamanho do volume solicitado no PVC.
`openshift_logging_es_pvc_prefix`	Um prefixo para os PVCs usados pelo serviço de Registro.
`openshift_logging_es_ops_pvc_dynamic`	Defina como `true` para usar volumes provisionados dinamicamente para a instância de log de operações.
`openshift_logging_es_ops_pvc_storage_class_name`	O nome da classe de armazenamento para a instância de log de operações.
`openshift_logging_es_ops_pvc_size`	O tamanho da solicitação de volume para a instância de operações.
`openshift_logging_es_ops_pvc_prefix`	Um prefixo para os PVCs de instância de OPS.

Implantar a pilha de logs

Se você estiver implantando o log como parte do processo inicial de instalação do OpenShift, então você só precisará seguir o processo de implantação padrão. O Ansible configurará e implantará os serviços necessários e os objetos OpenShift para que o serviço fique disponível assim que o Ansible for concluído.

No entanto, se você estiver realizando a implantação após a instalação inicial, o playbook do componente precisará ser usado pelo Ansible. Esse processo pode variar um pouco com diferentes versões do OpenShift,

portanto, certifique-se de ler e seguir "Documentação do Red Hat OpenShift Container Platform 3,11" para a sua versão.

Serviço de métricas

O serviço de métricas fornece informações valiosas ao administrador sobre o status, a utilização de recursos e a disponibilidade do cluster OpenShift. Também é necessário para a funcionalidade de escala automática de pods e muitas organizações usam dados do serviço de métricas para seus aplicativos de cobrança e/ou exibição.

Assim como no serviço de log e no OpenShift como um todo, o Ansible é usado para implantar o serviço de métricas. Além disso, tal como o serviço de registo, o serviço de métricas pode ser implementado durante uma configuração inicial do cluster ou após a sua operação utilizando o método de instalação do componente. As tabelas a seguir contêm as variáveis que são importantes ao configurar o armazenamento persistente para o serviço de métricas.

OBSERVAÇÃO	As tabelas abaixo contêm apenas as variáveis que são relevantes para a configuração de armazenamento, já que se refere ao serviço de métricas. Há muitas outras opções encontradas na documentação que devem ser revisadas, configuradas e usadas de acordo com sua implantação.

Variável	Detalhes
`openshift_metrics_storage_kind`	Defina como `nfs` para que o instalador crie um NFS PV para o serviço de log.
`openshift_metrics_storage_host`	O nome do host ou endereço IP do host NFS. Isso deve ser definido como dataLIF para o SVM.
`openshift_metrics_storage_nfs_directory`	O caminho de montagem para a exportação NFS. Por exemplo, se o volume for juntado como `/openshift_metrics`, você usaria esse caminho para essa variável.
`openshift_metrics_storage_volume_name`	O nome, por exemplo `pv_ose_metrics`, do PV a criar.
`openshift_metrics_storage_volume_size`	O tamanho da exportação NFS, por `100Gi` exemplo .

Se o cluster do OpenShift já estiver em execução e, portanto, o Trident tiver sido implantado e configurado, o instalador poderá usar o provisionamento dinâmico para criar os volumes. As variáveis a seguir precisarão ser configuradas.

Variável	Detalhes
`openshift_metrics_cassandra_pvc_prefix`	Um prefixo a ser usado para as PVCs de métricas.
`openshift_metrics_cassandra_pvc_size`	O tamanho dos volumes a solicitar.
`openshift_metrics_cassandra_storage_type`	O tipo de storage a ser usado para métricas, isso precisa ser definido como dinâmico para que o Ansible crie PVCs com a classe de storage apropriada.
`openshift_metrics_cassanda_pvc_storage_class_name`	O nome da classe de armazenamento a utilizar.

Implantar o serviço de métricas

Com as variáveis apropriadas do Ansible definidas no arquivo de hosts/inventário, implante o serviço com o Ansible. Se você estiver implantando no horário de instalação do OpenShift, o PV será criado e usado automaticamente. Se você estiver implantando usando os playbooks de componentes, após a instalação do OpenShift, o Ansible criará todos os PVCs necessários e, depois que o Trident provisionou o storage para eles, implantará o serviço.

As variáveis acima e o processo de implantação podem mudar a cada versão do OpenShift. Certifique-se de revisar e seguir "Guia de implantação do OpenShift da Red Hat" para a sua versão para que esteja configurado para o seu ambiente.

A proteção de dados e a recuperação de desastres do Trident

Saiba mais sobre as opções de proteção e recuperação para Trident e volumes criados usando o Trident. Você deve ter uma estratégia de proteção e recuperação de dados para cada aplicação com um requisito de persistência.

Replicação e recuperação do Trident

Você pode criar um backup para restaurar o Trident em caso de desastre.

Replicação Trident

O Trident usa CRDs do Kubernetes para armazenar e gerenciar seu próprio estado e o cluster etcd do Kubernetes para armazenar seus metadados.

Passos

1. Faça backup do cluster do Kubernetes etcd usando "Kubernetes: Fazer backup de um cluster etcd"o .
2. Coloque os artefatos de backup em um FlexVol volume

OBSERVAÇÃO	A NetApp recomenda a proteção do SVM em que o FlexVol reside, em uma relação da SnapMirror com outro SVM.

Recuperação de Trident

Usando CRDs do Kubernetes e o snapshot etcd do cluster do Kubernetes, você pode recuperar o Trident.

Passos

1. No SVM de destino, monte o volume que contém os arquivos de dados e certificados do Kubernetes no host, que será configurado como um nó mestre.
2. Copie todos os certificados necessários referentes ao cluster do Kubernetes `/etc/kubernetes/pki` e os arquivos de membros do etcd em `/var/lib/etcd`.
3. Restaure o cluster do Kubernetes a partir do backup etcd usando "Kubernetes: Restaurando um cluster etcd"o .
4. Execute `kubectl get crd` para verificar se todos os recursos personalizados do Trident foram criados e recuperar os objetos Trident para verificar se todos os dados estão disponíveis.

Replicação e recuperação da SVM

O Trident não pode configurar relacionamentos de replicação. No entanto, o administrador de storage pode usar "ONTAP SnapMirror" para replicar uma SVM.

Em caso de desastre, você pode ativar o SVM de destino do SnapMirror para começar a fornecer dados. Você pode voltar para o primário quando os sistemas são restaurados.

Sobre esta tarefa

Considere o seguinte ao usar o recurso de replicação do SnapMirror SVM:

- Você deve criar um back-end distinto para cada SVM com SVM-DR ativado.
- Configure as classes de storage para selecionar os back-ends replicados somente quando necessário, a fim de evitar ter volumes que não precisam de replicação provisionados nos back-ends compatíveis com SVM-DR.
- Os administradores de aplicativos devem entender o custo e a complexidade adicionais associados à replicação e considerar cuidadosamente seu plano de recuperação antes de iniciar esse processo.

Replicação da SVM

Você pode usar "ONTAP: Replicação do SnapMirror SVM" o para criar a relação de replicação do SVM.

O SnapMirror permite que você defina opções para controlar o que replicar. Você precisará saber quais opções você selecionou ao pré-formar Recuperação da SVM usando Trident.

- "-identidade-preservar verdadeiro" Replica toda a configuração da SVM.
- "-discard-configs network" Exclui LIFs e configurações de rede relacionadas.
- "-identity-preserve false" replica apenas os volumes e a configuração de segurança.

Recuperação da SVM usando Trident

O Trident não detecta falhas na SVM automaticamente. Em caso de desastre, o administrador pode iniciar manualmente o failover de Trident para a nova SVM.

Passos

1. Cancele transferências de SnapMirror agendadas e contínuas, interrompa a relação de replicação, pare o SVM de origem e ative o SVM de destino do SnapMirror.
2. Se você especificou `-identity-preserve false` ou `-discard-config network` ao configurar sua replicação SVM, atualize o `managementLIF` e `dataLIF` no arquivo de definição de back-end do Trident.
3. `storagePrefix`Confirmar está presente no arquivo de definição de back-end do Trident. Este parâmetro não pode ser alterado. Omitir `storagePrefix` fará com que a atualização de backend falhe.
4. Atualize todos os backends necessários para refletir o novo nome SVM de destino usando:

```
./tridentctl update backend <backend-name> -f <backend-json-file> -n
<namespace>
```

5. Se você especificou `-identity-preserve false` ou `discard-config network`, você deve rejeitar todos os pods de aplicativo.

OBSERVAÇÃO	Se você especificou `-identity-preserve true`, todos os volumes provisionados pelo Trident começam a fornecer dados quando o SVM de destino é ativado.

Replicação de volume e recuperação

O Trident não pode configurar as relações de replicação do SnapMirror. No entanto, o administrador de storage pode usar "Replicação e recuperação do ONTAP SnapMirror" para replicar volumes criados pelo Trident.

Em seguida, você pode importar os volumes recuperados para o Trident usando "importação de volume tridentctl"o .

OBSERVAÇÃO	A importação de volumes é compatível com o `ontap-san-economy` driver. A importação de Qtree com o `ontap-nas-economy` driver é um recurso de Tech Preview e não é compatível com o uso em produção. A importação não é compatível com o `ontap-flexgroup-economy` driver. Para obter detalhes, consulte "Importar volumes do Trident".

Proteção de dados do Snapshot

Você pode proteger e restaurar dados usando:

- Uma controladora de snapshot externa e CRDs para criar snapshots de volume do Kubernetes de volumes persistentes (PVS).

  "Instantâneos de volume"

- Snapshots ONTAP para restaurar todo o conteúdo de um volume ou recuperar arquivos individuais ou LUNs.

  "Snapshots ONTAP"

Automatizando o failover de aplicações com estado usando o Trident.

O recurso de desanexação forçada do Trident permite desanexar automaticamente volumes de nós com problemas em um cluster Kubernetes, evitando a corrupção de dados e garantindo a disponibilidade do aplicativo. Essa funcionalidade é particularmente útil em cenários onde os nós deixam de responder ou são desativados para manutenção.

Detalhes sobre Force Detach

A força de desprendimento está disponível para `ontap-san` , `ontap-san-economy` , `ontap-nas` , e `ontap-nas-economy` apenas. Antes de habilitar a desconexão forçada, o desligamento não normal do nó (NGNS) deve ser habilitado no cluster do Kubernetes. O NGNS é habilitado por padrão para o Kubernetes 1.28 e superior. Para mais informações, consulte "Kubernetes: Desligamento do nó não gracioso" .

OBSERVAÇÃO	Ao usar o `ontap-nas` driver ou `ontap-nas-economy`, você precisa definir o `autoExportPolicy` parâmetro na configuração de back-end `true` para que o

Trident possa restringir o acesso do nó Kubernetes com a alteração aplicada usando políticas de exportação gerenciadas.

AVISO

Como o Trident conta COM NGNS do Kubernetes, não remova `out-of-service` as taints de um nó que não seja saudável até que todos os workloads não toleráveis sejam reprogramados. Aplicar ou remover a taint de forma imprudente pode comprometer a proteção de dados no back-end.

Quando o administrador do cluster do Kubernetes tiver aplicado a `node.kubernetes.io/out-of-service=nodeshutdown:NoExecute` taint ao nó e `enableForceDetach` estiver definido como `true`, o Trident determinará o status do nó e:

1. Interrompa o acesso de E/S de backend para volumes montados nesse nó.
2. Marque o objeto do nó Trident como `dirty` (não é seguro para novas publicações).

OBSERVAÇÃO

O controlador Trident rejeitará novas solicitações de volume de publicação até que o nó seja requalificado (depois de ter sido marcado como `dirty`) pelo pod de nó do Trident. Quaisquer cargas de trabalho agendadas com um PVC montado (mesmo depois que o nó do cluster estiver pronto e saudável) não serão aceitas até que o Trident possa verificar o nó `clean` (seguro para novas publicações).

Quando a integridade do nó é restaurada e a taint é removida, o Trident irá:

1. Identifique e limpe caminhos publicados obsoletos no nó.
2. Se o nó estiver em um `cleanable` estado (a tint fora de serviço foi removida e o nó está `Ready` no estado) e todos os caminhos obsoletos e publicados estiverem limpos, o Trident reajustará o nó como `clean` e permitirá novos volumes publicados no nó.

Detalhes sobre o failover automático

Você pode automatizar o processo de desacoplamento forçado por meio da integração com"operador de verificação de integridade do nó (NHC)" . Quando ocorre uma falha em um nó, o NHC aciona a remediação de nó Trident (TNR) e a desconexão forçada automaticamente, criando um CR TridentNodeRemediation no namespace do Trident que define o nó com falha. O TNR é criado somente em caso de falha de um nó e removido pelo NHC assim que o nó volta a ficar online ou é excluído.

Falha no processo de remoção do pod do nó

O failover automático seleciona as cargas de trabalho a serem removidas do nó com falha. Quando um TNR é criado, o controlador TNR marca o nó como sujo, impedindo novas publicações de volume e inicia a remoção de pods compatíveis com o recurso de desanexação forçada e seus respectivos volumes anexados.

Todos os volumes/PVCs suportados por desanexação forçada também são suportados por failover automático:

- Volumes NAS e NAS-economy usando políticas de exportação automática (SMB ainda não é suportado).
- SAN e volumes SAN-economia.

ConsulteDetalhes sobre Force Detach .

Comportamento padrão:

- Os pods que utilizam volumes compatíveis com o recurso de desanexação forçada são removidos do nó com falha. O Kubernetes irá reprogramar essas tarefas para um nó saudável.
- Os pods que utilizam um volume não suportado pelo force-detach, incluindo volumes que não sejam Trident, não são removidos do nó com falha.
- Pods sem estado (não PVCs) não são removidos do nó com falha, a menos que a anotação do pod `trident.netapp.io/podRemediationPolicy: delete` Está definido.

Substituindo o comportamento de remoção do pod:

O comportamento de remoção de pods pode ser personalizado usando uma anotação de pod: `trident.netapp.io/podRemediationPolicy[retain, delete]` . Essas anotações são examinadas e utilizadas quando ocorre uma falha de sistema. Aplique anotações à especificação do pod do deployment/replicaset do Kubernetes para evitar que a anotação desapareça após uma falha:

- `retain`- O pod NÃO será removido do nó com falha durante um failover automático.
- `delete`- O Pod será removido do nó com falha durante um failover automático.

Essas anotações podem ser aplicadas a qualquer pod.

AVISO	• As operações de E/S serão bloqueadas apenas em nós com falha para volumes que suportam desanexação forçada. • Para volumes que não suportam desanexação forçada, existe o risco de corrupção de dados e problemas de multi-anexação.

Remediação de nó Trident CR

O CR TridentNodeRemediation (TNR) define um nó com falha. O nome do TNR é o nome do nó com falha.

Exemplo de TNR:

```
apiVersion: trident.netapp.io/v1
kind: TridentNodeRemediation
metadata:
  name: <K8s-node-name>
spec: {}
```

Estados TNR: Use os seguintes comandos para visualizar o status dos TNRs:
`kubectl get tnr <name> -n <trident-namespace>`

Os programas TNR (Captura, Esterilização e Devolução) podem ocorrer em um dos seguintes estados:

- *Remediando*:
  - Interrompa o acesso de E/S de backend para volumes suportados por desanexação forçada montados nesse nó.
  - O objeto de nó Trident está marcado como sujo (não é seguro para novas publicações).
  - Remova os pods e os anexos de volume do nó.

- *Recuperação de nó pendente*:
  - O controlador está aguardando que o nó volte a ficar online.
  - Assim que o nó estiver online, o comando `publish-enforcement` garantirá que o nó esteja limpo e pronto para novas publicações de volume.
- Se o nó for excluído do Kubernetes, o controlador TNR removerá o TNR e interromperá a reconciliação.
- *Concluído com sucesso*:
  - Todas as etapas de remediação e recuperação do nó foram concluídas com sucesso. O nó está limpo e pronto para novas publicações de volume.
- *Fracassado*:
  - Erro irrecuperável. Os motivos do erro são definidos no campo status.message da solicitação de alteração (CR).

Habilitando o failover automático

Pré-requisitos:

- Certifique-se de que a opção de desanexação forçada esteja ativada antes de ativar o failover automático. Para obter mais informações, consulteDetalhes sobre Force Detach .
- Instale a verificação de integridade do nó (NHC) no cluster Kubernetes.
  - "Instale o operator-sdk".
  - Instale o Operator Lifecycle Manager (OLM) no cluster, caso ainda não esteja instalado: `operator-sdk olm install` .
  - Instalar o operador de verificação de integridade do nó: `kubectl create -f https://operatorhub.io/install/node-healthcheck-operator.yaml` .

OBSERVAÇÃO	Você também pode usar métodos alternativos para detectar falha de nó, conforme especificado na Integrando soluções personalizadas de verificação de integridade de nós seção abaixo.

Ver"Operador de verificação de integridade do nó" para mais informações.

Passos

1. Crie um CR (Creative Request) NodeHealthCheck (NHC) no namespace Trident para monitorar os nós de trabalho no cluster. Exemplo:

```
apiVersion: remediation.medik8s.io/v1alpha1
kind: NodeHealthCheck
metadata:
  name: <CR name>
spec:
  selector:
    matchExpressions:
      - key: node-role.kubernetes.io/control-plane
        operator: DoesNotExist
      - key: node-role.kubernetes.io/master
```

```
        operator: DoesNotExist
  remediationTemplate:
    apiVersion: trident.netapp.io/v1
    kind: TridentNodeRemediationTemplate
    namespace: <Trident installation namespace>
    name: trident-node-remediation-template
  minHealthy: 0 # Trigger force-detach upon one or more node failures
  unhealthyConditions:
    - type: Ready
      status: "False"
      duration: 0s
    - type: Ready
      status: Unknown
      duration: 0s
```

2. Aplique o CR de verificação de integridade do nó no `trident` espaço de nomes.

```
kubectl apply -f <nhc-cr-file>.yaml -n <trident-namespace>
```

A CR acima está configurada para monitorar os nós de trabalho do Kubernetes em busca das condições de nó Ready: false e Unknown. O failover automático será acionado quando um nó entrar no estado Ready: false ou Ready: Unknown.

O `unhealthyConditions` No CR, utiliza-se um período de tolerância de 0 segundos. Isso faz com que o failover automático seja acionado imediatamente após o Kubernetes definir a condição do nó como Ready: false, que é definida depois que o Kubernetes perde o sinal de pulsação de um nó. O Kubernetes tem um tempo de espera padrão de 40 segundos após o último heartbeat antes de definir Ready: false. Esse período de tolerância pode ser personalizado nas opções de implantação do Kubernetes.

Para opções de configuração adicionais, consulte"Documentação do operador Node-Healthcheck" .

Informações adicionais de configuração

Quando o Trident é instalado com o recurso de desanexação forçada ativado, dois recursos adicionais são criados automaticamente no namespace do Trident para facilitar a integração com o NHC: TridentNodeRemediationTemplate (TNRT) e ClusterRole.

TridentNodeRemediationTemplate (TNRT):

O TNRT serve como modelo para o controlador NHC, que usa o TNRT para gerar recursos TNR conforme necessário.

```
apiVersion: trident.netapp.io/v1
kind: TridentNodeRemediationTemplate
metadata:
  name: trident-node-remediation-template
  namespace: trident
spec:
  template:
```

```
    spec: {}
```

Função do Cluster:

Uma função de cluster também é adicionada durante a instalação quando a opção de desanexação forçada está habilitada. Isso concede permissões ao NHC para TNRs no espaço de nomes Trident .

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  labels:
    rbac.ext-remediation/aggregate-to-ext-remediation: "true"
  name: tridentnoderemediation-access
rules:
- apiGroups:
  - trident.netapp.io
  resources:
  - tridentnoderemediationtemplates
  - tridentnoderemediations
  verbs:
  - get
  - list
  - watch
  - create
  - update
  - patch
  - delete
```

Atualizações e manutenção de clusters K8s

Para evitar falhas, pause o failover automático durante a manutenção ou atualizações do Kubernetes, períodos em que se espera que os nós sejam desligados ou reinicializados. Você pode pausar o CR do NHC (descrito acima) aplicando um patch em seu CR:

```
kubectl patch NodeHealthCheck <cr-name> --patch
'{"spec":{"pauseRequests":["<description-for-reason-of-pause>"]}}' --type=merge
```

Isso interrompe o failover automático. Para reativar o failover automático, remova o pauseRequests da especificação após a conclusão da manutenção.

Limitações

- As operações de E/S são impedidas apenas nos nós com falha para volumes suportados por desanexação forçada. Somente os pods que utilizam volumes/PVCs compatíveis com a função de desanexação forçada são removidos automaticamente.
- O failover automático e o desanexamento forçado são executados dentro do pod do controlador Trident. Se o nó que hospeda o trident-controller falhar, o failover automático será atrasado até que o Kubernetes mova o pod para um nó íntegro.

Integrando soluções personalizadas de verificação de integridade de nós

Você pode substituir o operador de verificação de integridade do nó por ferramentas alternativas de detecção de falhas de nó para acionar o failover automático. Para garantir a compatibilidade com o mecanismo de failover automático, sua solução personalizada deve:

- Crie um TNR quando uma falha de nó for detectada, usando o nome do nó com falha como o nome do CR do TNR.
- Exclua o TNR quando o nó se recuperar e o TNR estiver no estado "Sucesso".

Segurança

Recomendações de segurança do Trident

Use as recomendações listadas aqui para garantir que a instalação do Trident esteja segura.

Execute o Trident em seu próprio namespace

É importante impedir que aplicativos, administradores de aplicações, usuários e aplicativos de gerenciamento acessem as definições de objetos do Trident ou os pods para garantir um storage confiável e bloquear atividades maliciosas em potencial.

Para separar as outras aplicações e usuários do Trident, sempre instale o Trident em seu próprio namespace do Kubernetes (`trident`). A colocação do Trident em seu próprio namespace garante que somente o pessoal administrativo do Kubernetes tenha acesso ao pod Trident e aos artefatos (como segredos de back-end e CHAP, se aplicável) armazenados nos objetos CRD com namespaces. Você deve garantir que você permita que apenas administradores acessem o namespace Trident e, assim, acessem o `tridentctl` aplicativo.

Use a autenticação CHAP com backends ONTAP SAN

O Trident é compatível com autenticação baseada em CHAP para cargas de trabalho SAN ONTAP (usando os `ontap-san` drivers e `ontap-san-economy`). A NetApp recomenda o uso de CHAP bidirecional com Trident para autenticação entre um host e o back-end de storage.

Para backends ONTAP que usam os drivers de armazenamento SAN, o Trident pode configurar CHAP bidirecional e gerenciar nomes de usuário e segredos do CHAP através `tridentctl`do . "Prepare-se para configurar o back-end com drivers SAN ONTAP"Consulte para compreender como o Trident configura o CHAP nos backends ONTAP.

Use a autenticação CHAP com backends NetApp HCI e SolidFire

O NetApp recomenda a implantação de CHAP bidirecional para garantir a autenticação entre um host e os backends NetApp HCI e SolidFire. O Trident usa um objeto secreto que inclui duas senhas CHAP por locatário. Quando o Trident é instalado, ele gerencia os segredos CHAP e os armazena em um `tridentvolume` objeto CR para o respetivo PV. Quando você cria um PV, o Trident usa os segredos CHAP para iniciar uma sessão iSCSI e se comunicar com o sistema NetApp HCI e SolidFire através do CHAP.

OBSERVAÇÃO	Os volumes criados pelo Trident não estão associados a nenhum Grupo de Acesso por volume.

Use o Trident com NVE e NAE

O NetApp ONTAP fornece criptografia de dados em repouso para proteger dados confidenciais caso um disco seja roubado, retornado ou reutilizado. Para obter detalhes, "Configurar a visão geral da encriptação de volume do NetApp"consulte .

- Se o NAE estiver ativado no back-end, qualquer volume provisionado no Trident será habilitado para NAE.
  - Você pode definir o sinalizador de criptografia NVE como `""` para criar volumes habilitados para NAE.
- Se o NAE não estiver habilitado no back-end, qualquer volume provisionado no Trident será habilitado para NVE, a menos que o sinalizador de criptografia NVE esteja definido como `false` (o valor padrão) na configuração do back-end.

OBSERVAÇÃO	Os volumes criados no Trident em um back-end habilitado para NAE devem ser criptografados com NVE ou NAE. • Você pode definir o sinalizador de criptografia NVE como `true` na configuração de back-end do Trident para substituir a criptografia NAE e usar uma chave de criptografia específica por volume. • Definir o sinalizador de criptografia NVE como `false` em um back-end habilitado para NAE cria um volume habilitado para NAE. Não é possível desativar a criptografia NAE definindo o sinalizador de criptografia NVE como `false`.

- Você pode criar manualmente um volume NVE no Trident definindo explicitamente o sinalizador de criptografia NVE como `true`.

Para obter mais informações sobre opções de configuração de back-end, consulte:

- "Opções de configuração de SAN ONTAP"
- "Opções de configuração do ONTAP nas"

Configuração unificada de chave do Trident Linux (LUKS)

O Linux Unified Key Setup (LUKS) é uma especificação de criptografia de disco. O Trident usa o LUKS para fornecer criptografia por volume, no lado do host, para os volumes do ONTAP SAN e do ONTAP SAN ECONOMY, com suporte para rotação de senha e expansão de volume.

No Trident, os volumes criptografados por LUKS usam a cifra e o modo aes-xts-plain64, conforme recomendado "NIST"pelo .

OBSERVAÇÃO	A criptografia LUKS não é compatível com sistemas ASA r2. Para obter informações sobre sistemas ASA r2, consulte "Saiba mais sobre os sistemas de armazenamento ASA R2" .

Antes de começar

- Os nós de trabalho devem ter o cryptsetup 2,1 ou superior (mas inferior a 3,0) instalado. Para obter mais informações, visite "Gitlab: Cryptsetup".
- Por motivos de desempenho, a NetApp recomenda que os nós de trabalho suportem as novas instruções padrão de criptografia avançada (AES-NI). Para verificar o suporte ao AES-NI, execute o seguinte

comando:

```
grep "aes" /proc/cpuinfo
```

Se nada for devolvido, o processador não suporta AES-NI. Para obter mais informações sobre o AES-NI, visite: "Intel: Advanced Encryption Standard Instructions (AES-NI)".

Ativar encriptação LUKS

Você pode ativar a criptografia por volume no lado do host usando o LUKS (Configuração de chave unificada do Linux) para volumes ECONÔMICOS SAN ONTAP e SAN ONTAP.

Passos

1. Defina atributos de criptografia LUKS na configuração de back-end. Para obter mais informações sobre opções de configuração de back-end para SAN ONTAP, "Opções de configuração de SAN ONTAP"consulte .

```
{
  "storage": [
    {
      "labels": {
        "luks": "true"
      },
      "zone": "us_east_1a",
      "defaults": {
        "luksEncryption": "true"
      }
    },
    {
      "labels": {
        "luks": "false"
      },
      "zone": "us_east_1a",
      "defaults": {
        "luksEncryption": "false"
      }
    }
  ]
}
```

2. Use `parameters.selector` para definir os pools de armazenamento usando a criptografia LUKS. Por exemplo:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
```

```
  name: luks
provisioner: csi.trident.netapp.io
parameters:
  selector: "luks=true"
  csi.storage.k8s.io/node-stage-secret-name: luks-${pvc.name}
  csi.storage.k8s.io/node-stage-secret-namespace: ${pvc.namespace}
```

3. Crie um segredo que contenha a frase-passe LUKS. Por exemplo:

```
kubectl -n trident create -f luks-pvc1.yaml
apiVersion: v1
kind: Secret
metadata:
  name: luks-pvc1
stringData:
  luks-passphrase-name: A
  luks-passphrase: secretA
```

Use o LUKS com volumes de bloco bruto RWX

A partir do Trident 26.10, você pode usar a criptografia LUKS com volumes brutos de bloco com suporte a iSCSI que são anexados a vários nós do Kubernetes usando o `ReadWriteMany` modo de acesso (RWX). Isso é útil para cargas de trabalho, como as máquinas virtuais KubeVirt, que precisam anexar o mesmo volume criptografado em um nó de destino enquanto ele ainda está anexado ao nó de origem durante a migração ao vivo.

O suporte a RWX LUKS é limitado a volumes de bloco bruto. Volumes RWX LUKS em modo de sistema de arquivos não são suportados. A aplicação é responsável por coordenar o acesso simultâneo aos dados descriptografados; o Trident coordena apenas as operações LUKS.

Configure o LUKS no backend, na StorageClass e no Secret de passphrase, conforme descrito em Ativar encriptação LUKS. Em seguida, crie um PVC com `volumeMode: Block` e `ReadWriteMany`:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-rwx-pvc
  namespace: my-app
spec:
  accessModes:
    - ReadWriteMany
  volumeMode: Block
  storageClassName: san-luks-rwx
  resources:
    requests:
```

```
      storage: 10Gi
```

Pods em múltiplos nós podem conectar este PVC como um dispositivo de bloco bruto. O Trident formata o dispositivo com LUKS2 na primeira conexão e abre o volume criptografado existente em nós subsequentes. As operações LUKS existentes, incluindo rotação de senha e expansão de volume, têm suporte.

Por que o Trident serializa operações LUKS

O LUKS2 armazena cópias redundantes de seu cabeçalho de criptografia no início do dispositivo de bloco. `cryptsetup` pode reparar um cabeçalho quando detecta que as cópias diferem, inclusive durante comandos que parecem ser somente leitura. Seu bloqueio é local a um nó e não pode coordenar operações em outros nós. Se um nó estiver atualizando o cabeçalho enquanto outro executar `cryptsetup`, o segundo nó poderá acionar o reparo do cabeçalho e interromper a atualização, potencialmente corrompendo o cabeçalho.

Para evitar o acesso simultâneo ao cabeçalho, o Trident usa um Lease do Kubernetes para cada volume RWX LUKS. Antes que um nó execute qualquer `cryptsetup` comando para o volume, ele deve manter o Lease. O Trident libera o Lease após uma operação bem-sucedida. Se outro nó solicitar uma operação enquanto o Lease estiver em uso, o Trident retorna um erro passível de nova tentativa, e o Kubernetes tenta novamente mais tarde. As operações em volumes diferentes são independentes.

Se uma `cryptsetup` operação falhar, Trident retém intencionalmente o Lease para que apenas o mesmo nó possa tentar novamente. Um evento de aviso com o motivo `LeaseRetained` registra a falha. Os Leases não expiram automaticamente; um administrador deve remover um Lease somente após confirmar que nenhuma `cryptsetup` operação está em execução no nó detentor. Para obter instruções de recuperação, consulte "Corrija um Lease travado para volumes RWX do Linux Unified Key Setup (LUKS)".

Como os nós se revezam na abertura do volume, iniciar cargas de trabalho em vários nós diferentes pode demorar mais do que com um volume não criptografado. O Kubernetes tenta novamente erros de bloqueio transitórios após a conclusão da operação atual. Os pods podem permanecer em `ContainerCreating` por mais tempo durante grandes expansões; cargas de trabalho no mesmo nó podem compartilhar um único mapeamento de dispositivo aberto.

Limitações

Os volumes criptografados com LUKS não podem aproveitar a deduplicação e a compactação do ONTAP.

Configuração de back-end para importação de volumes LUKS

Para importar um volume LUKS, você deve definir `luksEncryption` como(`true` no back-end. A `luksEncryption` opção informa ao Trident se o volume é compatível com LUKS (`true`) ou não com LUKS (`false`), conforme mostrado no exemplo a seguir.

```
version: 1
storageDriverName: ontap-san
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
svm: trident_svm
username: admin
password: password
defaults:
  luksEncryption: 'true'
```

```
  spaceAllocation: 'false'
  snapshotPolicy: default
  snapshotReserve: '10'
```

Configuração de PVC para importação de volumes LUKS

Para importar volumes LUKS dinamicamente, defina a anotação `trident.netapp.io/luksEncryption` como `true` e inclua uma classe de armazenamento habilitada para LUKS no PVC, conforme mostrado neste exemplo.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: luks-pvc
  namespace: trident
  annotations:
    trident.netapp.io/luksEncryption: "true"
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: luks-sc
```

Rode uma frase-passe LUKS

Pode rodar a frase-passe LUKS e confirmar a rotação.

AVISO	Não se esqueça de uma frase-passe até ter verificado que ela não é mais referenciada por qualquer volume, instantâneo ou segredo. Se uma frase-passe referenciada for perdida, talvez você não consiga montar o volume e os dados permanecerão criptografados e inacessíveis.

Sobre esta tarefa

A rotação da frase-passe LUKS ocorre quando um pod que monta o volume é criado após uma nova frase-passe LUKS ser especificada. Quando um novo pod é criado, o Trident compara a frase-passe LUKS no volume com a frase-passe ativa no segredo.

- Se a frase-passe no volume não corresponder à frase-passe ativa no segredo, ocorre rotação.
- Se a frase-passe no volume corresponder à frase-passe ativa no segredo, o `previous-luks-passphrase` parâmetro é ignorado.

Passos

1. Adicione os `node-publish-secret-name` parâmetros e `node-publish-secret-namespace` StorageClass. Por exemplo:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: csi-san
provisioner: csi.trident.netapp.io
parameters:
  trident.netapp.io/backendType: "ontap-san"
  csi.storage.k8s.io/node-stage-secret-name: luks
  csi.storage.k8s.io/node-stage-secret-namespace: ${pvc.namespace}
  csi.storage.k8s.io/node-publish-secret-name: luks
  csi.storage.k8s.io/node-publish-secret-namespace: ${pvc.namespace}
```

2. Identificar senhas existentes no volume ou instantâneo.

Volume

```
tridentctl -d get volume luks-pvc1
GET http://127.0.0.1:8000/trident/v1/volume/<volumeID>

...luksPassphraseNames:["A"]
```

Snapshot

```
tridentctl -d get snapshot luks-pvc1
GET http://127.0.0.1:8000/trident/v1/volume/<volumeID>/<snapshotID>

...luksPassphraseNames:["A"]
```

3. Atualize o segredo LUKS para o volume para especificar as senhas novas e anteriores. Certifique-se `previous-luke-passphrase-name` e `previous-luks-passphrase` faça a correspondência da frase-passe anterior.

```
apiVersion: v1
kind: Secret
metadata:
  name: luks-pvc1
stringData:
  luks-passphrase-name: B
  luks-passphrase: secretB
  previous-luks-passphrase-name: A
  previous-luks-passphrase: secretA
```

4. Crie um novo pod de montagem do volume. Isto é necessário para iniciar a rotação.
5. Verifique se a senha foi girada.

Volume

```
tridentctl -d get volume luks-pvc1
GET http://127.0.0.1:8000/trident/v1/volume/<volumeID>

...luksPassphraseNames:["B"]
```

Snapshot

```
tridentctl -d get snapshot luks-pvc1
GET http://127.0.0.1:8000/trident/v1/volume/<volumeID>/<snapshotID>

...luksPassphraseNames:["B"]
```

Resultados

A frase-passe foi girada quando apenas a nova frase-passe é retornada no volume e no instantâneo.

OBSERVAÇÃO	Se duas senhas forem retornadas, por `luksPassphraseNames: ["B", "A"]` exemplo, a rotação estará incompleta. Você pode acionar um novo pod para tentar completar a rotação.

Ative a expansão de volume

Você pode ativar a expansão de volume em um volume criptografado com LUKS.

Passos

1. Ative a `CSINodeExpandSecret` porta de recurso (beta 1,25 ou mais). "Kubernetes 1,25: Use segredos para a expansão orientada por nós de volumes CSI"Consulte para obter detalhes.
2. Adicione os `node-expand-secret-name` parâmetros e `node-expand-secret-namespace` StorageClass. Por exemplo:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: luks
provisioner: csi.trident.netapp.io
parameters:
  selector: "luks=true"
  csi.storage.k8s.io/node-stage-secret-name: luks-${pvc.name}
  csi.storage.k8s.io/node-stage-secret-namespace: ${pvc.namespace}
  csi.storage.k8s.io/node-expand-secret-name: luks-${pvc.name}
  csi.storage.k8s.io/node-expand-secret-namespace: ${pvc.namespace}
allowVolumeExpansion: true
```

Resultados

Quando você inicia a expansão de armazenamento on-line, o kubelet passa as credenciais apropriadas para o driver.

Corrija um Lease travado para volumes RWX do Linux Unified Key Setup (LUKS)

Um volume ReadWriteMany (RWX) do Linux Unified Key Setup (LUKS) pode ficar bloqueado se um nó falhar enquanto estiver mantendo seu Lease do Kubernetes. Confirme que nenhuma `cryptsetup` operação ainda está em execução antes de excluir o Lease; excluí-lo durante uma operação pode colocar o cabeçalho LUKS em risco.

Entenda o lease

O Trident utiliza um Lease do Kubernetes para garantir que apenas um nó por vez execute `cryptsetup` comandos em um volume RWX LUKS. Os Leases são criados no namespace do Trident, e o nome do Lease é o nome do PersistentVolume (PV). A identidade do detentor identifica o nó que executa a operação.

O Trident exclui o Lease após uma operação bem-sucedida. Se uma operação falhar, o Trident retém o Lease para que apenas o mesmo nó possa tentar novamente. Os Leases não expiram nem são renovados automaticamente. Um evento de aviso com o motivo `LeaseRetained` inclui detalhes sobre uma operação com falha.

Identificar um Lease potencialmente travado

Os sintomas podem incluir pods presos em `ContainerCreating` ou `Terminating`, falha na expansão de volume ou erros transitórios de bloqueio.

1. Identifique o PVC usado pelo pod afetado:

```
kubectl get pod -n <app-namespace> <pod-name> \
  -o jsonpath='{.spec.volumes[*].persistentVolumeClaim.claimName}'
```

2. Encontre o PV associado ao PVC:

```
kubectl get pvc -n <app-namespace> <pvc-name> \
  -o jsonpath='{.spec.volumeName}'
```

3. Liste os Leases no namespace do Trident e identifique o Lease com o nome do PV:

```
kubectl get lease -n <trident-namespace>
```

4. Inspecione a Lease correspondente, incluindo o holder, a idade e os eventos:

```
kubectl describe lease -n <trident-namespace> <pv-name>
```

`cryptsetup` as operações normalmente são concluídas em segundos. Um lease com mais de 5 a 10

minutos pode indicar um deadlock. Verifique os `LeaseRetained` eventos em busca de erros repetidos, identifique o nó detentor do lease e verifique a integridade do nó:

```
kubectl get nodes
kubectl get pods -A -o wide --field-selector spec.nodeName=<lease-holder-
node>
```

Se o nó estiver íntegro e o Lease não apresentar eventos de aviso, inspecione os logs do pod do nó Trident no nó detentor antes de concluir que o Lease está travado:

```
kubectl get pods -n <trident-namespace> -o wide \
  --field-selector spec.nodeName=<lease-holder-node>
kubectl logs -n <trident-namespace> <trident-node-pod> -c trident-main
```

Se o problema subjacente no nó ou na rede puder ser corrigido, faça isso primeiro. A Lease pode ser resolvida quando o Trident concluir a operação com sucesso.

Resolva com segurança o Lease travado

IMPORTANTE	Não exclua um Lease enquanto o seu detentor ainda estiver executando uma `cryptsetup` operação. Comandos `cryptsetup` simultâneos de diferentes nós que não são coordenados pelo Lease podem corromper o cabeçalho LUKS.

Use a correção que corresponde à condição do nó:

- **O nó detentor falhou e foi desligado:** identifique os Leases mantidos por esse nó e exclua o Lease afetado. Se o nó se recuperar antes que suas cargas de trabalho sejam reprogramadas, permita que o Trident conclua sua operação e libere o Lease. Se as cargas de trabalho já tiverem sido movidas, aguarde até que seus pods no nó recuperado sejam totalmente encerrados, confirme se o deadlock persiste e, em seguida, remova o Lease obsoleto.
- **O nó detentor perdeu a conectividade com o plano de controle:** o nó ainda pode estar executando cargas de trabalho. Prefira desligar o nó antes de excluir o Lease. Se o nó permanecer acessível, pare o contêiner de nó do Trident nesse nó ou verifique nos logs se ele não consegue adquirir ou excluir o Lease e não está em execução `cryptsetup`. O Trident requer acesso à API do Kubernetes para adquirir o Lease antes de cada operação; após perder a conectividade, ele não pode iniciar outra operação. Aguarde pelo menos cinco minutos a partir da criação do Lease antes de considerar a exclusão, para permitir que qualquer comando em andamento seja concluído.
- **O nó detentor e a rede estão íntegros:** inspecione os eventos de aviso do Lease e os logs do nó do Trident para identificar a operação com falha. Se precisar mover a carga de trabalho afetada para outro nó, isole o nó detentor, exclua os pods do aplicativo afetados sem forçar a exclusão, aguarde até que sejam encerrados e remova o isolamento do nó. Exclua o Lease somente após confirmar que a operação original não está mais em execução.

Exclua o Lease somente depois de confirmar que é seguro fazer isso:

```
kubectl delete lease -n <trident-namespace> <pv-name>
```

Monitore os pods afetados e as operações de volume após a correção. Se o Lease for retido novamente, investigue o erro de cryptsetup relatado nos eventos do Lease e resolva o problema subjacente.

A criptografia Kerberos em trânsito do Trident

Usando a criptografia em trânsito Kerberos, você pode melhorar a segurança de acesso aos dados habilitando a criptografia para o tráfego entre o cluster gerenciado e o back-end de armazenamento.

O Trident oferece suporte à criptografia Kerberos para ONTAP como um back-end de armazenamento:

- **On-premise ONTAP** - Trident oferece suporte à criptografia Kerberos em conexões NFSv3 e NFSv4 de clusters Red Hat OpenShift e Kubernetes upstream para volumes ONTAP locais.

Você pode criar, excluir, redimensionar, snapshot, clone, clone somente leitura e importar volumes que usam criptografia NFS.

Configurar a criptografia Kerberos em trânsito com volumes ONTAP no local

Você pode ativar a criptografia Kerberos no tráfego de armazenamento entre o cluster gerenciado e um back-end de armazenamento ONTAP no local.

OBSERVAÇÃO	A criptografia Kerberos para tráfego NFS com backends de armazenamento ONTAP on-premise só é suportada usando o `ontap-nas` driver de armazenamento.

Antes de começar

- Certifique-se de que tem acesso ao `tridentctl` utilitário.
- Verifique se você tem acesso de administrador ao back-end de storage do ONTAP.
- Certifique-se de saber o nome do volume ou volumes que você compartilhará no back-end de storage do ONTAP.
- Certifique-se de que você preparou a VM de armazenamento ONTAP para oferecer suporte à criptografia Kerberos para volumes NFS. "Ative o Kerberos em um dataLIF"Consulte para obter instruções.
- Certifique-se de que todos os volumes NFSv4 que você usa com criptografia Kerberos estejam configurados corretamente. Consulte a seção Configuração de Domínio NFSv4 da NetApp (página 13) do "NetApp NFSv4 Melhorias e Guia de Práticas Recomendadas".

Adicionar ou modificar políticas de exportação do ONTAP

Você precisa adicionar regras às políticas de exportação existentes do ONTAP ou criar novas políticas de exportação que suportem a criptografia Kerberos para o volume raiz da VM de armazenamento do ONTAP, bem como quaisquer volumes do ONTAP compartilhados com o cluster do Kubernetes upstream. As regras de política de exportação que você adicionar ou as novas políticas de exportação que você criar precisam oferecer suporte aos seguintes protocolos de acesso e permissões de acesso:

Protocolos de acesso

Configure a política de exportação com os protocolos de acesso NFS, NFSv3 e NFSv4.

Aceder aos detalhes

Você pode configurar uma das três versões diferentes da criptografia Kerberos, dependendo de suas necessidades para o volume:

- **Kerberos 5** - (autenticação e criptografia)
- **Kerberos 5i** - (autenticação e criptografia com proteção de identidade)
- **Kerberos 5P** - (autenticação e criptografia com proteção de identidade e privacidade)

Configure a regra de política de exportação do ONTAP com as permissões de acesso apropriadas. Por exemplo, se os clusters estiverem montando os volumes NFS com uma mistura de criptografia Kerberos 5i e kerberos 5P, use as seguintes configurações de acesso:

Tipo	Acesso somente leitura	Acesso de leitura/escrita	Acesso ao superusuário
UNIX	Ativado	Ativado	Ativado
Kerberos 5i	Ativado	Ativado	Ativado
Kerberos 5P	Ativado	Ativado	Ativado

Consulte a documentação a seguir para saber como criar políticas de exportação e regras de política de exportação do ONTAP:

- "Crie uma política de exportação"
- "Adicione uma regra a uma política de exportação"

Crie um back-end de storage

Você pode criar uma configuração de back-end de armazenamento Trident que inclua o recurso de criptografia Kerberos.

Sobre esta tarefa

Quando você cria um arquivo de configuração de back-end de armazenamento que configura a criptografia Kerberos, você pode especificar uma das três versões diferentes da criptografia Kerberos usando o `spec.nfsMountOptions` parâmetro:

- `spec.nfsMountOptions: sec=krb5` (autenticação e criptografia)
- `spec.nfsMountOptions: sec=krb5i` (autenticação e criptografia com proteção de identidade)
- `spec.nfsMountOptions: sec=krb5p` (autenticação e criptografia com proteção de identidade e privacidade)

Especifique apenas um nível Kerberos. Se você especificar mais de um nível de criptografia Kerberos na lista de parâmetros, somente a primeira opção será usada.

Passos

1. No cluster gerenciado, crie um arquivo de configuração de back-end de storage usando o exemplo a seguir. Substitua os valores entre parêntesis> por informações do seu ambiente:

```
apiVersion: v1
kind: Secret
metadata:
  name: backend-ontap-nas-secret
type: Opaque
stringData:
```

```
    clientID: <CLIENT_ID>
    clientSecret: <CLIENT_SECRET>
---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-ontap-nas
spec:
  version: 1
  storageDriverName: "ontap-nas"
  managementLIF: <STORAGE_VM_MGMT_LIF_IP_ADDRESS>
  dataLIF: <PROTOCOL_LIF_FQDN_OR_IP_ADDRESS>
  svm: <STORAGE_VM_NAME>
  username: <STORAGE_VM_USERNAME_CREDENTIAL>
  password: <STORAGE_VM_PASSWORD_CREDENTIAL>
  nasType: nfs
  nfsMountOptions: ["sec=krb5i"] #can be krb5, krb5i, or krb5p
  qtreesPerFlexvol:
  credentials:
    name: backend-ontap-nas-secret
```

2. Use o arquivo de configuração que você criou na etapa anterior para criar o backend:

```
tridentctl create backend -f <backend-configuration-file>
```

Se a criação do backend falhar, algo está errado com a configuração do backend. Você pode exibir os logs para determinar a causa executando o seguinte comando:

```
tridentctl logs
```

Depois de identificar e corrigir o problema com o arquivo de configuração, você pode executar o comando create novamente.

Crie uma classe de armazenamento

Você pode criar uma classe de armazenamento para provisionar volumes com criptografia Kerberos.

Sobre esta tarefa

Ao criar um objeto de classe de armazenamento, você pode especificar uma das três versões diferentes da criptografia Kerberos usando o `mountOptions` parâmetro:

- `mountOptions: sec=krb5` (autenticação e criptografia)
- `mountOptions: sec=krb5i` (autenticação e criptografia com proteção de identidade)
- `mountOptions: sec=krb5p` (autenticação e criptografia com proteção de identidade e privacidade)

Especifique apenas um nível Kerberos. Se você especificar mais de um nível de criptografia Kerberos na lista de parâmetros, somente a primeira opção será usada. Se o nível de criptografia especificado na configuração de back-end de armazenamento for diferente do nível especificado no objeto de classe de armazenamento, o objeto de classe de armazenamento terá precedência.

Passos

1. Crie um objeto Kubernetes StorageClass, usando o exemplo a seguir:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-nas-sc
provisioner: csi.trident.netapp.io
mountOptions:
  - sec=krb5i #can be krb5, krb5i, or krb5p
parameters:
  backendType: ontap-nas
  storagePools: ontapnas_pool
  trident.netapp.io/nasType: nfs
allowVolumeExpansion: true
```

2. Crie a classe de armazenamento:

```
kubectl create -f sample-input/storage-class-ontap-nas-sc.yaml
```

3. Certifique-se de que a classe de armazenamento foi criada:

```
kubectl get sc ontap-nas-sc
```

   Você deve ver saída semelhante ao seguinte:

```
NAME          PROVISIONER             AGE
ontap-nas-sc    csi.trident.netapp.io   15h
```

Volumes de provisionamento

Depois de criar um back-end de storage e uma classe de storage, agora é possível provisionar um volume. Para obter instruções, "Provisionar um volume" consulte .

Configurar a criptografia Kerberos em trânsito com volumes Azure NetApp Files

Você pode ativar a criptografia Kerberos no tráfego de armazenamento entre o cluster gerenciado e um único back-end de armazenamento Azure NetApp Files ou um pool virtual de backends de armazenamento Azure NetApp Files.

Antes de começar

- Certifique-se de que você ativou o Trident no cluster gerenciado do Red Hat OpenShift.
- Certifique-se de que tem acesso ao `tridentctl` utilitário.
- Certifique-se de que preparou o back-end de armazenamento Azure NetApp Files para criptografia Kerberos, observando os requisitos e seguindo as instruções em "Documentação do Azure NetApp Files".
- Certifique-se de que todos os volumes NFSv4 que você usa com criptografia Kerberos estejam configurados corretamente. Consulte a seção Configuração de Domínio NFSv4 da NetApp (página 13) do "NetApp NFSv4 Melhorias e Guia de Práticas Recomendadas".

Crie um back-end de storage

Você pode criar uma configuração de back-end de armazenamento Azure NetApp Files que inclua o recurso de criptografia Kerberos.

Sobre esta tarefa

Quando você cria um arquivo de configuração de back-end de armazenamento que configura a criptografia Kerberos, você pode defini-lo para que ele seja aplicado em um dos dois níveis possíveis:

- O **nível de back-end de armazenamento** usando o `spec.kerberos` campo
- O **nível de pool virtual** usando o `spec.storage.kerberos` campo

Quando você define a configuração no nível do pool virtual, o pool é selecionado usando o rótulo na classe de armazenamento.

Em ambos os níveis, você pode especificar uma das três versões diferentes da criptografia Kerberos:

- `kerberos: sec=krb5` (autenticação e criptografia)
- `kerberos: sec=krb5i` (autenticação e criptografia com proteção de identidade)
- `kerberos: sec=krb5p` (autenticação e criptografia com proteção de identidade e privacidade)

Passos

1. No cluster gerenciado, crie um arquivo de configuração de back-end de storage usando um dos exemplos a seguir, dependendo de onde você precisa definir o back-end de storage (nível de back-end de armazenamento ou nível de pool virtual). Substitua os valores entre parêntesis> por informações do seu ambiente:

Exemplo de nível de back-end de storage

```
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-secret
type: Opaque
stringData:
  clientID: <CLIENT_ID>
  clientSecret: <CLIENT_SECRET>

---
```

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc
spec:
  version: 1
  storageDriverName: azure-netapp-files
  subscriptionID: <SUBSCRIPTION_ID>
  tenantID: <TENANT_ID>
  location: <AZURE_REGION_LOCATION>
  serviceLevel: Standard
  networkFeatures: Standard
  capacityPools: <CAPACITY_POOL>
  resourceGroups: <RESOURCE_GROUP>
  netappAccounts: <NETAPP_ACCOUNT>
  virtualNetwork: <VIRTUAL_NETWORK>
  subnet: <SUBNET>
  nasType: nfs
  kerberos: sec=krb5i #can be krb5, krb5i, or krb5p
  credentials:
    name: backend-tbc-secret
```

Exemplo de nível de pool virtual

```
---
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-secret
type: Opaque
stringData:
  clientID: <CLIENT_ID>
  clientSecret: <CLIENT_SECRET>

---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc
spec:
  version: 1
  storageDriverName: azure-netapp-files
  subscriptionID: <SUBSCRIPTION_ID>
  tenantID: <TENANT_ID>
  location: <AZURE_REGION_LOCATION>
  serviceLevel: Standard
```

```
    networkFeatures: Standard
    capacityPools: <CAPACITY_POOL>
    resourceGroups: <RESOURCE_GROUP>
    netappAccounts: <NETAPP_ACCOUNT>
    virtualNetwork: <VIRTUAL_NETWORK>
    subnet: <SUBNET>
    nasType: nfs
    storage:
      - labels:
          type: encryption
        kerberos: sec=krb5i #can be krb5, krb5i, or krb5p
    credentials:
      name: backend-tbc-secret
```

2. Use o arquivo de configuração que você criou na etapa anterior para criar o backend:

```
tridentctl create backend -f <backend-configuration-file>
```

Se a criação do backend falhar, algo está errado com a configuração do backend. Você pode exibir os logs para determinar a causa executando o seguinte comando:

```
tridentctl logs
```

Depois de identificar e corrigir o problema com o arquivo de configuração, você pode executar o comando create novamente.

Crie uma classe de armazenamento

Você pode criar uma classe de armazenamento para provisionar volumes com criptografia Kerberos.

Passos

1. Crie um objeto Kubernetes StorageClass, usando o exemplo a seguir:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: sc-nfs
provisioner: csi.trident.netapp.io
parameters:
  backendType: azure-netapp-files
  trident.netapp.io/nasType: nfs
  selector: type=encryption
```

2. Crie a classe de armazenamento:

```
kubectl create -f sample-input/storage-class-sc-nfs.yaml
```

3. Certifique-se de que a classe de armazenamento foi criada:

```
kubectl get sc -sc-nfs
```

Você deve ver saída semelhante ao seguinte:

```
NAME          PROVISIONER             AGE
sc-nfs        csi.trident.netapp.io   15h
```

Volumes de provisionamento

Depois de criar um back-end de storage e uma classe de storage, agora é possível provisionar um volume. Para obter instruções, "Provisionar um volume" consulte .

Proteja seus aplicativos com o Trident Protect.

Saiba mais sobre o Trident Protect.

O NetApp Trident Protect oferece recursos avançados de gerenciamento de dados de aplicativos que aprimoram a funcionalidade e a disponibilidade de aplicativos Kubernetes com estado, com suporte dos sistemas de armazenamento NetApp ONTAP e do provisionador de armazenamento NetApp Trident CSI. O Trident Protect simplifica o gerenciamento, a proteção e a movimentação de cargas de trabalho conteinerizadas em nuvens públicas e ambientes locais. Ele também oferece recursos de automação por meio de sua API e CLI.

Você pode proteger aplicativos com o Trident Protect criando recursos personalizados (CRs) ou usando a CLI do Trident Protect.

O que se segue?

Você pode consultar os requisitos do Trident Protect antes de instalá-lo:

- "Requisitos do Trident Protect"

Instale o Trident Protect

Requisitos do Trident Protect

Comece verificando se seu ambiente operacional, clusters de aplicativos, aplicativos e licenças estão prontos. Certifique-se de que seu ambiente atenda a esses requisitos para implantar e operar o Trident Protect.

Compatibilidade do cluster Kubernetes com o Trident Protect

O Trident Protect é compatível com uma ampla gama de ofertas de Kubernetes totalmente gerenciadas e autogerenciadas, incluindo:

- Amazon Elastic Kubernetes Service (EKS)
- Google Kubernetes Engine (GKE)
- Microsoft Azure Kubernetes Service (AKS)
- Red Hat OpenShift
- SUSE Harvester 1.7.0 (ONTAP iSCSI)
- SUSE Rancher
- Portfólio do VMware Tanzu
- Kubernetes upstream

OBSERVAÇÃO	• Os backups do Trident Protect são suportados apenas em nós de computação Linux. Os nós de computação do Windows não são suportados para operações de backup.

- Certifique-se de que o cluster no qual você instala o Trident Protect esteja configurado com um controlador de snapshots em execução e os CRDs relacionados. Para instalar um controlador de snapshots, consulte "estas instruções" .
- Certifique-se de que exista pelo menos um VolumeSnapshotClass. Para mais informações, consulte "VolumeSnapshotClass".
- É necessário o Helm 4.x ou posterior para instalar Trident Protect.

Compatibilidade com o backend de armazenamento Trident Protect

O Trident Protect é compatível com os seguintes sistemas de armazenamento:

- Amazon FSX para NetApp ONTAP
- Cloud Volumes ONTAP
- Storage arrays ONTAP
- Google Cloud NetApp volumes
- Azure NetApp Files

Certifique-se de que o back-end de storage atenda aos seguintes requisitos:

- Certifique-se de que o armazenamento NetApp conectado ao cluster esteja usando o Trident 24.02 ou mais recente (Trident 24.10 é recomendado).
- Verifique se você tem um back-end de storage do NetApp ONTAP.
- Certifique-se de ter configurado um bucket de armazenamento de objetos para armazenar backups.
- Crie todos os namespaces de aplicativos que você planeja usar para aplicativos ou operações de gerenciamento de dados de aplicativos. O Trident Protect não cria esses namespaces para você; se você especificar um namespace inexistente em um recurso personalizado, a operação falhará.

Requisitos para volumes nas-economia

O Trident Protect oferece suporte a operações de backup e restauração para volumes NAS Economy. Atualmente, não há suporte para snapshots, clones e replicação do SnapMirror para volumes nas-economy. Você precisa habilitar um diretório de snapshots para cada volume nas-economy que planeja usar com o Trident Protect.

OBSERVAÇÃO

Alguns aplicativos não são compatíveis com volumes que usam um diretório instantâneo. Para esses aplicativos, você precisa ocultar o diretório instantâneo executando o seguinte comando no sistema de armazenamento ONTAP:

```
nfs modify -vserver <svm> -v3-hide-snapshot enabled
```

Você pode ativar o diretório de snapshot executando o seguinte comando para cada volume de economia nas, substituindo `<volume-UUID>` pelo UUID do volume que deseja alterar:

```
tridentctl update volume <volume-UUID> --snapshot-dir=true --pool-level
=true -n trident
```

OBSERVAÇÃO	Você pode habilitar diretórios de snapshot por padrão para novos volumes definindo a opção de configuração de back-end do Trident `snapshotDir` como `true`. Os volumes existentes não são afetados.

Proteção de dados com máquinas virtuais do KubeVirt

O Trident Protect oferece recursos de congelamento e descongelamento do sistema de arquivos para máquinas virtuais KubeVirt durante operações de proteção de dados, garantindo a consistência dos dados. O método de configuração e o comportamento padrão para operações de congelamento de máquinas virtuais variam entre as versões do Trident Protect, sendo que as versões mais recentes oferecem configuração simplificada por meio de parâmetros do gráfico Helm.

OBSERVAÇÃO	Durante as operações de restauração, qualquer `VirtualMachineSnapshots` criados para uma máquina virtual (VM) não são restaurados.

▼ **Trident Protect 25.10 e versões mais recentes**

O Trident Protect congela e descongela automaticamente os sistemas de arquivos do KubeVirt durante as operações de proteção de dados para garantir a consistência. A partir do Trident Protect 25.10, você pode desativar esse comportamento usando o `vm.freeze` parâmetro durante a instalação do gráfico Helm. O parâmetro está ativado por padrão.

```
helm install ... --set vm.freeze=false ...
```

▼ **Trident Protect 24.10.1 a 25.06**

A partir da versão 24.10.1 do Trident Protect, o sistema congela e descongela automaticamente os sistemas de arquivos do KubeVirt durante as operações de proteção de dados. Opcionalmente, você pode desativar esse comportamento automático usando o seguinte comando:

```
kubectl set env deployment/trident-protect-controller-manager
NEPTUNE_VM_FREEZE=false -n trident-protect
```

▼ **Trident Protect 24.10**

O Trident Protect 24.10 não garante automaticamente um estado consistente para os sistemas de arquivos das VMs do KubeVirt durante as operações de proteção de dados. Se você deseja proteger os dados da sua máquina virtual KubeVirt usando o Trident Protect 24.10, precisa habilitar manualmente a funcionalidade de congelamento/descongelamento dos sistemas de arquivos antes da operação de proteção de dados. Isso garante que os sistemas de arquivos estejam em um estado consistente.

Você pode configurar Trident Protect 24.10 para gerenciar o congelamento e descongelamento do sistema de arquivos da VM durante as operações de proteção de dados "configuração da virtualização" e, em seguida, usar o seguinte comando:

```
kubectl set env deployment/trident-protect-controller-manager
NEPTUNE_VM_FREEZE=true -n trident-protect
```

Para aprender como definir aplicativos e restaurar VMs com Trident Protect, consulte "Proteja máquinas virtuais KubeVirt com Trident Protect".

Requisitos para replicação do SnapMirror

A replicação NetApp SnapMirror está disponível para uso com o Trident Protect para as seguintes soluções ONTAP :

- Sistemas NetApp FAS, AFF e ASA locais. A replicação SnapMirror com Trident protect não é atualmente suportada para sistemas ASA r2.
- NetApp ONTAP Select
- NetApp Cloud Volumes ONTAP
- Amazon FSX para NetApp ONTAP

Requisitos de cluster do ONTAP para replicação do SnapMirror

Se você planeja usar a replicação do SnapMirror, verifique se o cluster do ONTAP atende aos seguintes requisitos:

- * NetApp Trident*: O NetApp Trident deve estar presente nos clusters Kubernetes de origem e destino que utilizam o ONTAP como backend. O Trident Protect oferece suporte à replicação com a tecnologia NetApp SnapMirror usando classes de armazenamento com suporte dos seguintes drivers:
  - `ontap-nas` : NFS
  - `ontap-san` : iSCSI
  - `ontap-san` :FC
  - `ontap-san` : NVMe/TCP (requer no mínimo a versão ONTAP 9.15.1)
- **Licenças**: As licenças assíncronas do ONTAP SnapMirror usando o pacote proteção de dados devem estar ativadas nos clusters ONTAP de origem e destino. "Visão geral do licenciamento do SnapMirror no ONTAP"Consulte para obter mais informações.

  A partir do ONTAP 9.10,1, todas as licenças são entregues como um arquivo de licença NetApp (NLF), que é um único arquivo que permite vários recursos. "Licenças incluídas no ONTAP One"Consulte para obter mais informações.

OBSERVAÇÃO	Somente a proteção assíncrona SnapMirror é suportada.

Considerações de peering para replicação do SnapMirror

Certifique-se de que seu ambiente atenda aos seguintes requisitos se você planeja usar peering de back-end de storage:

- **Cluster e SVM**: Os backends de storage do ONTAP devem ser colocados em Contato. "Visão geral do peering de cluster e SVM"Consulte para obter mais informações.

OBSERVAÇÃO	Certifique-se de que os nomes do SVM usados na relação de replicação entre dois clusters ONTAP sejam exclusivos.

- **NetApp Trident e SVM**: Os SVMs remotos pareados devem estar disponíveis para o NetApp Trident no cluster de destino.
- **Backends gerenciados**: Você precisa adicionar e gerenciar backends de armazenamento ONTAP no Trident Protect para criar uma relação de replicação.

Configuração Trident / ONTAP para replicação SnapMirror

O Trident Protect exige que você configure pelo menos um backend de armazenamento que suporte replicação para os clusters de origem e destino. Se os clusters de origem e destino forem os mesmos, o aplicativo de destino deverá usar um backend de armazenamento diferente do aplicativo de origem para obter a melhor resiliência.

Requisitos de cluster do Kubernetes para replicação do SnapMirror

Certifique-se de que seus clusters do Kubernetes atendam aos seguintes requisitos:

- **Acessibilidade do AppVault**: Os clusters de origem e destino devem ter acesso à rede para ler e gravar no AppVault para replicação de objetos do aplicativo.
- **Conectividade de rede**: configure regras de firewall, permissões de bucket e listas de permissões de IP para permitir a comunicação entre os dois clusters e o AppVault através de WANs.

OBSERVAÇÃO	Muitos ambientes corporativos implementam políticas rígidas de firewall em conexões WAN. Verifique esses requisitos de rede com sua equipe de infraestrutura antes de configurar a replicação.

Instale e configure o Trident Protect.

Se o seu ambiente atender aos requisitos do Trident Protect, você pode seguir estas etapas para instalar o Trident Protect em seu cluster. Você pode obter o Trident Protect da NetApp ou instalá-lo a partir do seu próprio registro privado. A instalação a partir de um registro privado é útil caso seu cluster não tenha acesso à Internet.

Instale o Trident Protect

Instale o Trident Protect da NetApp.

Passos

1. Adicione o repositório Helm do Trident:

```
helm repo add netapp-trident-protect
https://netapp.github.io/trident-protect-helm-chart
```

2. Use o Helm para instalar o Trident Protect. Substituir `<name-of-cluster>` com um nome de cluster, que será atribuído ao cluster e usado para identificar os backups e snapshots do cluster:

```
helm install trident-protect netapp-trident-protect/trident-protect
--set clusterName=<name-of-cluster> --version 100.2610.0 --create
-namespace --namespace trident-protect
```

3. Opcionalmente, para ativar o registro de depuração (recomendado para resolução de problemas), use:

```
helm install trident-protect netapp-trident-protect/trident-protect
--set clusterName=<name-of-cluster> --set logLevel=debug --version
100.2610.0 --create-namespace --namespace trident-protect
```

O registro de depuração ajuda o suporte da NetApp a solucionar problemas sem exigir alterações no nível de registro ou reprodução do problema.

Instale o Trident Protect a partir de um registro privado.

Você pode instalar o Trident Protect a partir de um registro de imagens privado caso seu cluster Kubernetes não tenha acesso à Internet. Nestes exemplos, substitua os valores entre colchetes por informações do seu ambiente:

Passos

1. Puxe as seguintes imagens para a sua máquina local, atualize as etiquetas e, em seguida, envie-as para o seu registo privado:

```
docker.io/netapp/controller:26.10.1
docker.io/netapp/trident-protect-utils:v6.0.0
docker.io/netapp/trident-protect-guestfs:v2.0.0
docker.io/netapp/kopiablockrestore:26.10.1
docker.io/netapp/trident-autosupport:26.10.1
docker.io/netapp/exechook:26.10.1
docker.io/netapp/resourcebackup:26.10.1
docker.io/netapp/resourcerestore:26.10.1
docker.io/netapp/resourcedelete:26.10.1
docker.io/netapp/filebrowse-transfer:26.10.1
```

Por exemplo:

```
docker pull docker.io/netapp/controller:26.10.1
```

```
docker tag docker.io/netapp/controller:26.10.1 <private-registry-
url>/controller:26.10.1
```

```
docker push <private-registry-url>/controller:26.10.1
```

OBSERVAÇÃO	Para obter o Helm chart, primeiro baixe o Helm chart em uma máquina com acesso à internet usando `helm pull trident-protect --version 100.2610.0 --repo https://netapp.github.io/trident-protect-helm-chart`, depois copie o arquivo resultante `trident-protect-100.2610.0.tgz` para seu ambiente offline e instale usando `helm install trident-protect ./trident-`

`protect-100.2610.0.tgz` em vez da referência ao repositório na etapa final.

2. Crie o namespace do sistema Trident Protect:

```
kubectl create ns trident-protect
```

3. Inicie sessão no registo:

```
helm registry login <private-registry-url> -u <account-id> -p <api-token>
```

4. Crie um segredo para usar para autenticação de Registro privado:

```
kubectl create secret docker-registry regcred --docker-username=<registry-username> --docker-password=<api-token> -n trident-protect --docker-server=<private-registry-url>
```

5. Adicione o repositório Helm do Trident:

```
helm repo add netapp-trident-protect https://netapp.github.io/trident-protect-helm-chart
```

6. Crie um arquivo chamado `protectValues.yaml` . Certifique-se de que contenha as seguintes configurações do Trident Protect:

```
---
imageRegistry: <private-registry-url>
imagePullSecrets:
  - name: regcred
```

OBSERVAÇÃO

O `imageRegistry` e `imagePullSecrets` Os valores se aplicam a todas as imagens componentes, incluindo `resourcebackup` e `resourcerestore` . Se você enviar imagens para um caminho de repositório específico dentro do seu registro (por exemplo, `example.com:443/my-repo`), inclua o caminho completo no campo de registro. Isso garantirá que todas as imagens sejam extraídas de `<private-registry-url>/<image-name>:<tag>`.

7. Use o Helm para instalar o Trident Protect. Substituir `<name_of_cluster>` com um nome de cluster, que será atribuído ao cluster e usado para identificar os backups e snapshots do cluster:

```
helm install trident-protect netapp-trident-protect/trident-protect
--set clusterName=<name_of_cluster> --version 100.2610.0 --create
-namespace --namespace trident-protect -f protectValues.yaml
```

8. Opcionalmente, para ativar o registro de depuração (recomendado para resolução de problemas), use:

```
helm install trident-protect netapp-trident-protect/trident-protect
--set clusterName=<name-of-cluster> --set logLevel=debug --version
100.2610.0 --create-namespace --namespace trident-protect -f
protectValues.yaml
```

O registro de depuração ajuda o suporte da NetApp a solucionar problemas sem exigir alterações no nível de registro ou reprodução do problema.

OBSERVAÇÃO Para opções adicionais de configuração do gráfico Helm, incluindo configurações de AutoSupport e filtragem de namespace, consulte "Personalize a instalação do Trident Protect".

Instale o plugin Trident Protect CLI

Você pode usar o plugin de linha de comando Trident Protect, que é uma extensão do Trident. `tridentctl` Utilitário para criar e interagir com recursos personalizados (CRs) do Trident Protect.

Instale o plugin Trident Protect CLI

Antes de usar o utilitário de linha de comando, você precisa instalá-lo na máquina usada para acessar o cluster. Siga estes passos, dependendo se a sua máquina utiliza uma CPU x64 ou ARM.

Faça o download do plugin para CPUs Linux AMD64

Passos

1. Baixe o plugin Trident Protect CLI:

```
curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-
protect/releases/download/26.10.1/tridentctl-protect-linux-amd64
```

Faça o download do plugin para CPUs Linux ARM64

Passos

1. Baixe o plugin Trident Protect CLI:

```
curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-
```

```
protect/releases/download/26.10.1/tridentctl-protect-linux-arm64
```

Baixe o plugin para CPUs Mac AMD64

Passos

1. Baixe o plugin Trident Protect CLI:

```
curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-protect/releases/download/26.10.1/tridentctl-protect-macos-amd64
```

Baixe o plugin para CPUs Mac ARM64

Passos

1. Baixe o plugin Trident Protect CLI:

```
curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-protect/releases/download/26.10.1/tridentctl-protect-macos-arm64
```

1. Ativar permissões de execução para o binário do plugin:

```
chmod +x tridentctl-protect
```

2. Copie o binário do plugin para um local definido na variável PATH. Por exemplo, `/usr/bin` ou `/usr/local/bin` (você pode precisar de Privileges elevado):

```
cp ./tridentctl-protect /usr/local/bin/
```

3. Opcionalmente, você pode copiar o binário do plugin para um local em seu diretório home. Neste caso, é recomendável garantir que a localização faça parte da variável PATH:

```
cp ./tridentctl-protect ~/bin/
```

OBSERVAÇÃO Copiar o plugin para um local em sua variável PATH permite que você use o plugin digitando `tridentctl-protect` ou `tridentctl protect` de qualquer local.

Veja a ajuda do plugin Trident CLI

Você pode usar os recursos integrados de ajuda do plugin para obter ajuda detalhada sobre os recursos do plugin:

Passos

1. Utilize a função de ajuda para visualizar as orientações de utilização:

```
tridentctl-protect help
```

Ativar a auto-conclusão do comando

Após instalar o plugin Trident Protect CLI, você pode ativar o recurso de autocompletar para determinados comandos.

Ative a auto-conclusão para o shell Bash

Passos

1. Crie o script de conclusão:

```
tridentctl-protect completion bash > tridentctl-completion.bash
```

2. Crie um novo diretório em seu diretório inicial para conter o script:

```
mkdir -p ~/.bash/completions
```

3. Mova o script baixado para `~/.bash/completions` o diretório:

```
mv tridentctl-completion.bash ~/.bash/completions/
```

4. Adicione a seguinte linha ao `~/.bashrc` arquivo em seu diretório inicial:

```
source ~/.bash/completions/tridentctl-completion.bash
```

Ative a auto-conclusão para o shell Z.

Passos

1. Crie o script de conclusão:

```
tridentctl-protect completion zsh > tridentctl-completion.zsh
```

2. Crie um novo diretório em seu diretório inicial para conter o script:

```
mkdir -p ~/.zsh/completions
```

3. Mova o script baixado para `~/.zsh/completions` o diretório:

```
mv tridentctl-completion.zsh ~/.zsh/completions/
```

4. Adicione a seguinte linha ao `~/.zprofile` arquivo em seu diretório inicial:

```
source ~/.zsh/completions/tridentctl-completion.zsh
```

Resultado

Após o seu próximo login shell, você pode usar o comando auto-completação com o plugin tridentctl-protect.

Personalize a instalação do Trident Protect

Você pode personalizar a configuração padrão do Trident Protect para atender aos requisitos específicos do seu ambiente.

Especifique os limites de recursos do contêiner Trident Protect.

Você pode usar um arquivo de configuração para especificar limites de recursos para os contêineres do Trident Protect após a instalação do Trident Protect. A definição de limites de recursos permite controlar a quantidade de recursos do cluster que são consumidos pelas operações do Trident Protect.

Passos

1. Crie um arquivo chamado `resourceLimits.yaml`.
2. Preencha o arquivo com as opções de limite de recursos para os contêineres do Trident Protect de acordo com as necessidades do seu ambiente.

   O seguinte exemplo de arquivo de configuração mostra as configurações disponíveis e contém os valores padrão para cada limite de recursos:

```
---
jobResources:
  defaults:
    limits:
      cpu: 8000m
      memory: 10000Mi
      ephemeralStorage: ""
    requests:
      cpu: 100m
      memory: 100Mi
      ephemeralStorage: ""
  resticVolumeBackup:
    limits:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
    requests:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
```

```
  resticVolumeRestore:
    limits:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
    requests:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
  kopiaVolumeBackup:
    limits:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
    requests:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
  kopiaVolumeRestore:
    limits:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
    requests:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
```

3. Aplique os valores do `resourceLimits.yaml` arquivo:

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect -f resourceLimits.yaml --reuse-values
```

Personalizar restrições de contexto de segurança

Você pode usar um arquivo de configuração para modificar as restrições de contexto de segurança (SCCs) do OpenShift para contêineres Trident Protect após instalar o Trident Protect. Essas restrições definem as limitações de segurança para os pods em um cluster Red Hat OpenShift.

Passos

1. Crie um arquivo chamado `sccconfig.yaml`.
2. Adicione a opção SCC ao arquivo e modifique os parâmetros de acordo com as necessidades do seu ambiente.

   O exemplo a seguir mostra os valores padrão dos parâmetros para a opção SCC:

```
scc:
  create: true
  name: trident-protect-job
  priority: 1
```

Esta tabela descreve os parâmetros para a opção SCC:

Parâmetro	Descrição	Padrão
criar	Determina se um recurso SCC pode ser criado. Um recurso SCC será criado somente se `scc.create` estiver definido como `true` e o processo de instalação Helm identificar um ambiente OpenShift. Se não estiver operando no OpenShift, ou se `scc.create` estiver definido como `false`, nenhum recurso SCC será criado.	verdadeiro
nome	Especifica o nome do SCC.	Trident-protect-job
prioridade	Define a prioridade do SCC. Os SCCs com valores de prioridade mais elevados são avaliados antes daqueles com valores mais baixos.	1

3. Aplique os valores do `sccconfig.yaml` arquivo:

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect -f sccconfig.yaml --reuse-values
```

Isso substituirá os valores padrão pelos especificados no `sccconfig.yaml` arquivo.

Configure as definições adicionais do gráfico de navegação do Trident Protect.

Você pode personalizar as configurações do AutoSupport, o filtro de namespace e o gerenciamento de certificados de webhook do OpenShift para atender às suas necessidades específicas. A tabela a seguir descreve os parâmetros de configuração disponíveis:

Parâmetro	Tipo	Descrição
autoSupport.proxy	cadeia de carateres	Configura um URL de proxy para conexões do NetApp AutoSupport . Use isso para rotear uploads de pacotes de suporte por meio de um servidor proxy. Exemplo: `http://my.proxy.url` .

Parâmetro	Tipo	Descrição
autoSupport.inseguro	booleano	Ignora a verificação TLS para conexões proxy AutoSupport quando definido como `true` . Use somente para conexões proxy inseguras. (padrão: `false`)
autoSupport.habilitado	booleano	Ativa ou desativa os uploads diários do pacote Trident Protect AutoSupport . Quando definido para `false` Os uploads diários agendados estão desativados, mas você ainda pode gerar pacotes de suporte manualmente. (padrão: `true`)
restaurarSkipNamespaceAnnotations	cadeia de carateres	Lista separada por vírgulas de anotações de namespace a serem excluídas das operações de backup e restauração. Permite filtrar namespaces com base em anotações.
restaurarIgnorarEtiquetasDeEspaçoDeNomes	cadeia de carateres	Lista separada por vírgulas de rótulos de namespace a serem excluídos das operações de backup e restauração. Permite filtrar namespaces com base em rótulos.
useOpenshiftServiceCA	booleano	Habilita a integração com o operador de CA de serviço nativo do OpenShift para criação e injeção automática de certificados TLS para o webhook de admissão do Trident Protect. Quando definido como `true`, o operador de CA de serviço gerencia automaticamente os certificados TLS do webhook. Use somente em clusters OpenShift. (padrão: `false`)

Para criptografar os metadados de snapshot e backup, consulte "Criptografe os metadados do AppVault para snapshots e backups do Trident Protect".

Você pode configurar essas opções usando um arquivo de configuração YAML ou sinalizadores de linha de comando:

Usar arquivo YAML

Passos

1. Crie um arquivo de configuração e nomeie-o `values.yaml` .
2. No arquivo que você criou, adicione as opções de configuração que deseja personalizar.

```
autoSupport:
  enabled: false
  proxy: http://my.proxy.url
  insecure: true
restoreSkipNamespaceAnnotations: "annotation1,annotation2"
restoreSkipNamespaceLabels: "label1,label2"
useOpenshiftServiceCA: true
```

3. Depois de preencher o `values.yaml` arquivo com os valores corretos, aplique o arquivo de configuração:

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect -f values.yaml --reuse-values
```

Usar sinalizador CLI

Passos

1. Use o seguinte comando com o `--set` sinalizador para especificar parâmetros individuais:

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect \
  --set autoSupport.enabled=false \
  --set autoSupport.proxy=http://my.proxy.url \
  --set-string
restoreSkipNamespaceAnnotations="{annotation1,annotation2}" \
  --set-string restoreSkipNamespaceLabels="{label1,label2}" \
  --set useOpenshiftServiceCA=true \
  --reuse-values
```

Restrinja os pods do Trident Protect a nós específicos.

Você pode usar a restrição de seleção de nós `nodeSelector` do Kubernetes para controlar quais de seus nós são elegíveis para executar pods do Trident Protect, com base nos rótulos dos nós. Por padrão, o Trident Protect é restrito a nós que executam Linux. Você pode personalizar ainda mais essas restrições de acordo com suas necessidades.

Passos

1. Crie um arquivo chamado `nodeSelectorConfig.yaml`.
2. Adicione a opção nodeSeletor ao arquivo e modifique o arquivo para adicionar ou alterar rótulos de nó para restringir de acordo com as necessidades do seu ambiente. Por exemplo, o arquivo a seguir contém a restrição padrão do sistema operacional, mas também tem como alvo uma região específica e nome do aplicativo:

```
nodeSelector:
```

```
    kubernetes.io/os: linux
    region: us-west
    app.kubernetes.io/name: mysql
```

3. Aplique os valores do `nodeSelectorConfig.yaml` arquivo:

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect -f nodeSelectorConfig.yaml --reuse-values
```

Isso substitui as restrições padrão com as que você especificou no `nodeSelectorConfig.yaml` arquivo.

Gerenciar Trident Protect

Gerencie a autorização e o controle de acesso do Trident Protect.

O Trident Protect utiliza o modelo Kubernetes de controle de acesso baseado em funções (RBAC). Por padrão, o Trident Protect fornece um único namespace de sistema e sua respectiva conta de serviço padrão. Se a sua organização possui muitos usuários ou necessidades específicas de segurança, você pode usar os recursos RBAC do Trident Protect para obter um controle mais granular sobre o acesso a recursos e namespaces.

O administrador do cluster sempre tem acesso a recursos no namespace padrão `trident-protect` e também pode acessar recursos em todos os outros namespaces. Para controlar o acesso a recursos e aplicações, é necessário criar espaços de nomes adicionais e adicionar recursos e aplicações a esses espaços de nomes.

Observe que nenhum usuário pode criar CRS de gerenciamento de dados do aplicativo no namespace padrão `trident-protect`. Você precisa criar CRS de gerenciamento de dados de aplicativo em um namespace de aplicativo (como prática recomendada, criar CRS de gerenciamento de dados de aplicativo no mesmo namespace que seu aplicativo associado).

OBSERVAÇÃO	Somente os administradores devem ter acesso aos objetos de recursos personalizados privilegiados do Trident Protect, que incluem: • **AppVault**: Requer dados de credenciais de bucket • **Pacote de Suporte Automático**: Coleta métricas, registros e outros dados confidenciais do Trident Protect. • **AutoSupportBundleSchedule**: Gerencia os horários de coleta de Registros Como prática recomendada, use o RBAC para restringir o acesso a objetos privilegiados aos administradores.

Para obter mais informações sobre como o RBAC regula o acesso a recursos e namespaces, consulte o "Documentação do Kubernetes RBAC".

Para obter informações sobre contas de serviço, consulte o "Documentação da conta de serviço do Kubernetes".

Exemplo: Gerencie o acesso para dois grupos de usuários

Por exemplo, uma organização tem um administrador de cluster, um grupo de usuários de engenharia e um grupo de usuários de marketing. O administrador do cluster concluiria as seguintes tarefas para criar um ambiente onde o grupo de engenharia e o grupo de marketing tenham acesso apenas aos recursos atribuídos aos respetivos namespaces.

Etapa 1: Crie um namespace para conter recursos para cada grupo

Criar um namespace permite separar recursos logicamente e controlar melhor quem tem acesso a esses recursos.

Passos

1. Crie um namespace para o grupo de engenharia:

```
kubectl create ns engineering-ns
```

2. Crie um namespace para o grupo de marketing:

```
kubectl create ns marketing-ns
```

Etapa 2: Crie novas contas de serviço para interagir com recursos em cada namespace

Cada novo namespace que você criar vem com uma conta de serviço padrão, mas você deve criar uma conta de serviço para cada grupo de usuários para que você possa dividir ainda mais Privileges entre grupos no futuro, se necessário.

Passos

1. Crie uma conta de serviço para o grupo de engenharia:

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: eng-user
  namespace: engineering-ns
```

2. Crie uma conta de serviço para o grupo de marketing:

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: mkt-user
  namespace: marketing-ns
```

Passo 3: Crie um segredo para cada nova conta de serviço

Um segredo de conta de serviço é usado para autenticar com a conta de serviço e pode ser facilmente excluído e recriado se comprometido.

Passos

1. Crie um segredo para a conta de serviço de engenharia:

```
apiVersion: v1
kind: Secret
metadata:
  annotations:
    kubernetes.io/service-account.name: eng-user
  name: eng-user-secret
  namespace: engineering-ns
type: kubernetes.io/service-account-token
```

2. Crie um segredo para a conta do serviço de marketing:

```
apiVersion: v1
kind: Secret
metadata:
  annotations:
    kubernetes.io/service-account.name: mkt-user
  name: mkt-user-secret
  namespace: marketing-ns
type: kubernetes.io/service-account-token
```

Passo 4: Crie um objeto RoleBinding para vincular o objeto ClusterRole a cada nova conta de serviço

Um objeto ClusterRole padrão é criado quando você instala o Trident Protect. Você pode vincular essa ClusterRole à conta de serviço criando e aplicando um objeto RoleBinding.

Passos

1. Vincule o ClusterRole à conta de serviço de engenharia:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: engineering-ns-tenant-rolebinding
  namespace: engineering-ns
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: trident-protect-tenant-cluster-role
subjects:
```

```
- kind: ServiceAccount
  name: eng-user
  namespace: engineering-ns
```

2. Vincule o ClusterRole à conta do serviço de marketing:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: marketing-ns-tenant-rolebinding
  namespace: marketing-ns
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: trident-protect-tenant-cluster-role
subjects:
- kind: ServiceAccount
  name: mkt-user
  namespace: marketing-ns
```

Passo 5: Testar permissões

Teste se as permissões estão corretas.

Passos

1. Confirme se os usuários de engenharia podem acessar os recursos de engenharia:

```
kubectl auth can-i --as=system:serviceaccount:engineering-ns:eng-user
get applications.protect.trident.netapp.io -n engineering-ns
```

2. Confirme que os usuários de engenharia não podem acessar recursos de marketing:

```
kubectl auth can-i --as=system:serviceaccount:engineering-ns:eng-user
get applications.protect.trident.netapp.io -n marketing-ns
```

Etapa 6: Conceder acesso a objetos AppVault

Para executar tarefas de gerenciamento de dados, como backups e snapshots, o administrador do cluster precisa conceder acesso a objetos AppVault a usuários individuais.

Passos

1. Crie e aplique um arquivo YAML de combinação secreta e AppVault que concede a um usuário acesso a um AppVault. Por exemplo, o CR a seguir concede acesso a um AppVault ao usuário `eng-user`:

```
apiVersion: v1
data:
  accessKeyID: <ID_value>
  secretAccessKey: <key_value>
kind: Secret
metadata:
  name: appvault-for-eng-user-only-secret
  namespace: trident-protect
type: Opaque
---
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: appvault-for-eng-user-only
  namespace: trident-protect # Trident Protect system namespace
spec:
  providerConfig:
    azure:
      accountName: ""
      bucketName: ""
      endpoint: ""
    gcp:
      bucketName: ""
      projectID: ""
    s3:
      bucketName: testbucket
      endpoint: 192.168.0.1:30000
      secure: "false"
      skipCertValidation: "true"
  providerCredentials:
    accessKeyID:
      valueFromSecret:
        key: accessKeyID
        name: appvault-for-eng-user-only-secret
    secretAccessKey:
      valueFromSecret:
        key: secretAccessKey
        name: appvault-for-eng-user-only-secret
  providerType: GenericS3
```

2. Crie e aplique um CR de função para permitir que os administradores de cluster concedam acesso a recursos específicos em um namespace. Por exemplo:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
```

```
metadata:
  name: eng-user-appvault-reader
  namespace: trident-protect
rules:
- apiGroups:
  - protect.trident.netapp.io
  resourceNames:
  - appvault-for-enguser-only
  resources:
  - appvaults
  verbs:
  - get
```

3. Criar e aplicar um RoleBinding CR para vincular as permissões ao usuário eng-user. Por exemplo:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: eng-user-read-appvault-binding
  namespace: trident-protect
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: eng-user-appvault-reader
subjects:
- kind: ServiceAccount
  name: eng-user
  namespace: engineering-ns
```

4. Verifique se as permissões estão corretas.

   a. Tente recuperar informações de objeto AppVault para todos os namespaces:

   ```
   kubectl get appvaults -n trident-protect
   --as=system:serviceaccount:engineering-ns:eng-user
   ```

   Você deve ver saída semelhante ao seguinte:

   ```
   Error from server (Forbidden): appvaults.protect.trident.netapp.io is
   forbidden: User "system:serviceaccount:engineering-ns:eng-user"
   cannot list resource "appvaults" in API group
   "protect.trident.netapp.io" in the namespace "trident-protect"
   ```

b. Teste para ver se o usuário pode obter as informações do AppVault que ele agora tem permissão para acessar:

```
kubectl auth can-i --as=system:serviceaccount:engineering-ns:eng-user
get appvaults.protect.trident.netapp.io/appvault-for-eng-user-only -n
trident-protect
```

Você deve ver saída semelhante ao seguinte:

```
yes
```

Resultado

Os usuários aos quais você concedeu permissões AppVault devem poder usar objetos AppVault autorizados para operações de gerenciamento de dados de aplicativos e não devem poder acessar recursos fora dos namespaces atribuídos ou criar novos recursos aos quais eles não têm acesso.

Monitorar recursos do Trident Protect

Você pode usar as ferramentas de código aberto kube-state-metrics, Prometheus e Alertmanager para monitorar a integridade dos recursos protegidos pelo Trident Protect.

O serviço kube-state-metrics gera métricas a partir da comunicação da API do Kubernetes. Ao utilizá-lo com o Trident Protect, você obtém informações úteis sobre o estado dos recursos em seu ambiente.

O Prometheus é um conjunto de ferramentas que pode ingerir os dados gerados pelo kube-state-metrics e apresentá-los como informações facilmente legíveis sobre esses objetos. Em conjunto, o kube-state-metrics e o Prometheus oferecem uma maneira de monitorar a integridade e o status dos recursos que você está gerenciando com o Trident Protect.

Alertmanager é um serviço que ingere os alertas enviados por ferramentas como Prometheus e os encaminha para destinos que você configura.

OBSERVAÇÃO

As configurações e orientações incluídas nessas etapas são apenas exemplos; você precisa personalizá-las para corresponder ao seu ambiente. Consulte a seguinte documentação oficial para obter instruções e suporte específicos:

- "documentação de métricas de estado do kube"
- "Documentação do Prometheus"
- "Documentação do Alertmanager"

Passo 1: Instale as ferramentas de monitoramento

Para habilitar o monitoramento de recursos no Trident Protect, você precisa instalar e configurar o kube-state-metrics, o Promethus e o Alertmanager.

Instalar métricas de estado do kube

Você pode instalar métricas de estado do kube usando o Helm.

Passos

1. Adicione o gráfico Helm de métricas de estado kube. Por exemplo:

```
helm repo add prometheus-community https://prometheus-
community.github.io/helm-charts
helm repo update
```

2. Aplique o Prometheus ServiceMonitor CRD ao cluster:

```
kubectl apply -f https://raw.githubusercontent.com/prometheus-
operator/prometheus-operator/main/example/prometheus-operator-
crd/monitoring.coreos.com_servicemonitors.yaml
```

3. Crie um arquivo de configuração para o gráfico Helm (por exemplo, `metrics-config.yaml`). Você pode personalizar o seguinte exemplo de configuração para corresponder ao seu ambiente:

Metrics-config.yaml: Configuração do gráfico Helm do kube-State-metrics

```
---
extraArgs:
  # Collect only custom metrics
  - --custom-resource-state-only=true

customResourceState:
  enabled: true
  config:
    kind: CustomResourceStateMetrics
    spec:
      resources:
      - groupVersionKind:
          group: protect.trident.netapp.io
          kind: "Backup"
          version: "v1"
        labelsFromPath:
          backup_uid: [metadata, uid]
          backup_name: [metadata, name]
          creation_time: [metadata, creationTimestamp]
        metrics:
        - name: backup_info
          help: "Exposes details about the Backup state"
          each:
            type: Info
            info:
              labelsFromPath:
                appVaultReference: ["spec", "appVaultRef"]
                appReference: ["spec", "applicationRef"]
```

```
rbac:
  extraRules:
  - apiGroups: ["protect.trident.netapp.io"]
    resources: ["backups"]
    verbs: ["list", "watch"]

# Collect metrics from all namespaces
namespaces: ""

# Ensure that the metrics are collected by Prometheus
prometheus:
  monitor:
    enabled: true
```

4. Instale as métricas de estado do kube implantando o gráfico Helm. Por exemplo:

```
helm install custom-resource -f metrics-config.yaml prometheus-
community/kube-state-metrics --version 5.21.0
```

5. Configure o kube-state-metrics para gerar métricas para os recursos personalizados usados pelo Trident Protect, seguindo as instruções em "documentação de recursos personalizados de métricas de estado do kube" .

Instale Prometheus

Você pode instalar o Prometheus seguindo as instruções no "Documentação do Prometheus".

Instale o Alertmanager

Você pode instalar o Alertmanager seguindo as instruções no "Documentação do Alertmanager".

Passo 2: Configure as ferramentas de monitoramento para trabalhar em conjunto

Depois de instalar as ferramentas de monitoramento, você precisa configurá-las para trabalhar em conjunto.

Passos

1. Integre o kube-State-metrics com Prometheus. Edite o arquivo de configuração Prometheus (`prometheus.yaml`) e adicione as informações do serviço kube-State-metrics. Por exemplo:

   prometheus.yaml: integração do serviço kube-state-metrics com o Prometheus

```
---
apiVersion: v1
kind: ConfigMap
metadata:
  name: prometheus-config
  namespace: trident-protect
data:
```

```
  prometheus.yaml: |
    global:
      scrape_interval: 15s
    scrape_configs:
      - job_name: 'kube-state-metrics'
        static_configs:
          - targets: ['kube-state-metrics.trident-protect.svc:8080']
```

2. Configure Prometheus para rotear alertas para Alertmanager. Edite o arquivo de configuração Prometheus (`prometheus.yaml`) e adicione a seguinte seção:

prometheus.yaml: Enviar alertas para o Alertmanager

```
alerting:
  alertmanagers:
    - static_configs:
        - targets:
            - alertmanager.trident-protect.svc:9093
```

Resultado

Prometheus agora pode coletar métricas de kube-State-metrics e enviar alertas para Alertmanager. Agora você está pronto para configurar quais condições acionam um alerta e onde os alertas devem ser enviados.

Etapa 3: Configurar alertas e destinos de alerta

Depois de configurar as ferramentas para trabalhar em conjunto, você precisa configurar que tipo de informação aciona alertas e para onde os alertas devem ser enviados.

Exemplo de alerta: Falha de backup

O exemplo a seguir define um alerta crítico que é acionado quando o status do recurso personalizado de backup é definido como `Error` por 5 segundos ou mais. Você pode personalizar este exemplo para corresponder ao seu ambiente e incluir esse snippet YAML em seu `prometheus.yaml` arquivo de configuração:

rules.yaml: Defina um alerta do Prometheus para backups com falha

```
rules.yaml: |
  groups:
    - name: fail-backup
        rules:
          - alert: BackupFailed
            expr: kube_customresource_backup_info{status="Error"}
            for: 5s
            labels:
              severity: critical
            annotations:
              summary: "Backup failed"
```

```
            description: "A backup has failed."
```

Configure o Alertmanager para enviar alertas para outros canais

Você pode configurar o Alertmanager para enviar notificações para outros canais, como e-mail, PagerDuty, Microsoft Teams ou outros serviços de notificação especificando a respetiva configuração no `alertmanager.yaml` arquivo.

O exemplo a seguir configura o Alertmanager para enviar notificações para um canal do Slack. Para personalizar este exemplo para o ambiente, substitua o valor da `api_url` chave pelo URL do webhook do Slack usado no ambiente:

alertmanager.yaml: Enviar alertas para um canal do Slack

```
data:
  alertmanager.yaml: |
    global:
      resolve_timeout: 5m
    route:
      receiver: 'slack-notifications'
    receivers:
      - name: 'slack-notifications'
        slack_configs:
          - api_url: '<your-slack-webhook-url>'
            channel: '#failed-backups-channel'
            send_resolved: false
```

Gere um pacote de suporte Trident Protect

O Trident Protect permite que os administradores gerem pacotes que incluem informações úteis para o Suporte da NetApp , como logs, métricas e informações de topologia sobre os clusters e aplicativos gerenciados. Se você estiver conectado à Internet, poderá carregar pacotes de suporte no NetApp Support Site (NSS) usando um arquivo de recurso personalizado (CR).

Crie um pacote de suporte usando um CR

Passos

1. Crie o arquivo de recurso personalizado (CR) e nomeie-o (por exemplo, `trident-protect-support-bundle.yaml`).
2. Configure os seguintes atributos:
   - **metadata.name**: (*required*) o nome deste recurso personalizado; escolha um nome único e sensível para o seu ambiente.
   - **Spec.triggerType**: (*required*) determina se o pacote de suporte é gerado imediatamente ou programado. A geração de pacotes programados acontece às 12AM UTC. Valores possíveis:
     - Programado

        - Manual
    - **Spec.uploadEnabled**: (*Optional*) controla se o pacote de suporte deve ser carregado para o site de suporte da NetApp depois que ele é gerado. Se não for especificado, o padrão é `false`. Valores possíveis:
        - verdadeiro
        - falso (padrão)
    - **Spec.dataWindowStart**: (*Optional*) Uma cadeia de carateres de data no formato RFC 3339 que especifica a data e a hora em que a janela de dados incluídos no pacote de suporte deve começar. Se não for especificado, o padrão é 24 horas atrás. A data da janela mais antiga que você pode especificar é de 7 dias atrás.

      Exemplo YAML:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: AutoSupportBundle
metadata:
  name: trident-protect-support-bundle
spec:
  triggerType: Manual
  uploadEnabled: true
  dataWindowStart: 2024-05-05T12:30:00Z
```

3. Depois de preencher o `trident-protect-support-bundle.yaml` arquivo com os valores corretos, aplique o CR:

```
kubectl apply -f trident-protect-support-bundle.yaml -n trident-
protect
```

Crie um pacote de suporte usando a CLI

Passos

1. Crie o pacote de suporte, substituindo valores entre parênteses por informações do seu ambiente. O `trigger-type` determina se o pacote é criado imediatamente ou se o tempo de criação é ditado pelo agendamento e pode ser `Manual` ou `Scheduled`. A predefinição é `Manual`.

   Por exemplo:

```
tridentctl-protect create autosupportbundle <my-bundle-name>
--trigger-type <trigger-type> -n trident-protect
```

Monitore e recupere o pacote de suporte

Depois de criar um pacote de suporte usando qualquer um dos métodos, você pode monitorar seu progresso de geração e recuperá-lo para seu sistema local.

Passos

1. Espere pelo `status.generationState` alcançar `Completed` estado. Você pode monitorar o progresso da geração com o seguinte comando:

```
kubectl get autosupportbundle trident-protect-support-bundle -n trident-protect
```

2. Recupere o pacote de suporte para seu sistema local. Obtenha o comando de cópia do pacote AutoSupport concluído:

```
kubectl describe autosupportbundle trident-protect-support-bundle -n trident-protect
```

   Encontre o `kubectl cp` comando da saída e execute-o, substituindo o argumento de destino pelo seu diretório local preferido.

Aprimore o Trident Protect

Você pode atualizar o Trident Protect para a versão mais recente para aproveitar os novos recursos ou correções de bugs.

OBSERVAÇÃO

- Ao atualizar a partir da versão 24.10, os snapshots executados durante a atualização podem falhar. Essa falha não impede a criação de snapshots futuros, sejam eles manuais ou agendados. Se um snapshot falhar durante a atualização, você pode criar um novo snapshot manualmente para garantir a proteção do seu aplicativo.

  Para evitar possíveis falhas, você pode desabilitar todos os agendamentos de snapshots antes da atualização e reabilitá-los posteriormente. No entanto, isso resultará na perda de snapshots agendados durante o período de atualização.

- Para instalações em registros privados, certifique-se de que o gráfico Helm e as imagens necessárias para a versão desejada estejam disponíveis em seu registro privado e verifique se seus valores Helm personalizados são compatíveis com a nova versão do gráfico. Para obter mais informações, consulte"Instale o Trident Protect a partir de um registro privado." .

Passo 1: Selecione uma versão

As versões do Trident Protect seguem uma convenção de nomenclatura baseada em datas `YY.MM`, onde "YY" são os dois últimos dígitos do ano e "MM" é o mês. As versões com ponto seguem uma `YY.MM.X` convenção, onde "X" é o nível de patch. Você selecionará a versão para atualizar com base na versão da qual está atualizando.

- Você pode realizar uma atualização direta para qualquer versão de destino que esteja dentro de um intervalo de quatro versões da sua versão instalada. Por exemplo, você pode atualizar diretamente de 24.10 (ou qualquer versão secundária de 24.10) para 25.10.
- Se você estiver atualizando de uma versão fora do período de quatro versões, execute uma atualização

em várias etapas. Use as instruções de atualização para a "versão anterior" da qual você está atualizando para atualizar para a versão mais recente que se encaixe no período de quatro versões. Por exemplo, se você estiver executando a versão 24.10 e quiser atualizar para a 26.10:

a. Primeira atualização da 24.10 para a 25.06.
b. Em seguida, atualize de 25.06 para 26.10.

Etapa 2: atualize Trident Protect

Para atualizar o Trident Protect, siga os passos abaixo.

Passos

1. Atualize o repositório Helm do Trident:

```
helm repo update
```

2. Atualize os CRDs do Trident Protect:

OBSERVAÇÃO	Esta etapa é necessária se você estiver atualizando de uma versão anterior à 25.06, pois os CRDs agora estão incluídos no gráfico Helm do Trident Protect.

a. Execute este comando para mudar o gerenciamento de CRDs de `trident-protect-crds` para `trident-protect` :

```
kubectl get crd | grep protect.trident.netapp.io | awk '{print $1}' |
xargs -I {} kubectl patch crd {} --type merge -p '{"metadata":
{"annotations":{"meta.helm.sh/release-name": "trident-protect"}}}'
```

b. Execute este comando para excluir o segredo do Helm para o `trident-protect-crds` gráfico:

OBSERVAÇÃO	Não desinstale o `trident-protect-crds` gráfico usando o Helm, pois isso pode remover seus CRDs e quaisquer dados relacionados.

```
kubectl delete secret -n trident-protect -l name=trident-protect-
crds,owner=helm
```

3. Aprimore o Trident Protect:

```
helm upgrade trident-protect netapp-trident-protect/trident-protect
--version 100.2610.0 --namespace trident-protect
```

OBSERVAÇÃO	Você pode configurar o nível de registro durante a atualização adicionando `--set logLevel=debug` ao comando de atualização. O nível de registro padrão é `warn` . O registro de depuração é recomendado para a resolução de problemas, pois ajuda o suporte da NetApp a diagnosticar problemas sem exigir

alterações no nível de registro ou reprodução do problema.

Gerenciar e proteger aplicativos

Use objetos do Trident Protect AppVault para gerenciar buckets.

O recurso personalizado (CR) do bucket para o Trident Protect é conhecido como AppVault. Os objetos do AppVault são a representação declarativa do fluxo de trabalho do Kubernetes de um bucket de armazenamento. Um AppVault CR contém as configurações necessárias para que um bucket seja usado em operações de proteção, como backups, snapshots, operações de restauração e replicação do SnapMirror . Somente administradores podem criar AppVaults.

Você precisa criar uma CR do AppVault manualmente ou pela linha de comando ao executar operações de proteção de dados em um aplicativo. A CR do AppVault é específica para o seu ambiente, e você pode usar os exemplos nesta página como guia para criar CRs do AppVault.

OBSERVAÇÃO	Certifique-se de que o AppVault CR esteja no cluster onde o Trident Protect está instalado. Se o AppVault CR não existir ou você não conseguir acessá-lo, a linha de comando exibirá um erro.

Configurar a autenticação e as senhas do AppVault

Antes de criar um AppVault CR, certifique-se de que o AppVault e o movimentador de dados escolhido possam ser autenticados com o provedor e quaisquer recursos relacionados.

Senhas do repositório do controlador de dados

Ao criar objetos AppVault usando CRs ou o plugin Trident Protect CLI, você pode especificar um segredo do Kubernetes com senhas personalizadas para criptografia Restic e Kopia. Se você não especificar uma senha secreta, o Trident Protect usará uma senha padrão.

- Ao criar manualmente CRs do AppVault, use o campo **spec.dataMoverPasswordSecretRef** para especificar o segredo.
- Ao criar objetos do AppVault usando a CLI do Trident Protect, utilize o `--data-mover-password-secret-ref` argumento para especificar o segredo.

OBSERVAÇÃO	Para criptografar os metadados de snapshot e backup ou para alterar a senha do cofre, use o `spec.vaultPasswordSecretRef` campo. Consulte "Criptografe os metadados do AppVault para snapshots e backups do Trident Protect".

Crie um segredo de senha do repositório do mover de dados

Utilize os exemplos a seguir para criar a senha secreta. Ao criar objetos do AppVault, você pode instruir o Trident Protect a usar esse segredo para autenticar com o repositório de movimentação de dados.

OBSERVAÇÃO	• Dependendo do motor de dados que você está usando, você só precisa incluir a senha correspondente para esse controlador de dados. Por exemplo, se você estiver usando Restic e não planeja usar o Kopia no futuro, você pode incluir apenas a senha Restic quando criar o segredo.

- Guarde a senha em um local seguro. Você precisará dela para restaurar dados no mesmo cluster ou em um diferente. Se o cluster ou o `trident-protect` namespace for excluído, você não poderá restaurar seus backups ou snapshots sem a senha.

Use um CR

```
---
apiVersion: v1
data:
  KOPIA_PASSWORD: <base64-encoded-password>
  RESTIC_PASSWORD: <base64-encoded-password>
kind: Secret
metadata:
  name: my-optional-data-mover-secret
  namespace: trident-protect
type: Opaque
```

Use a CLI

```
kubectl create secret generic my-optional-data-mover-secret \
--from-literal=KOPIA_PASSWORD=<plain-text-password> \
--from-literal=RESTIC_PASSWORD=<plain-text-password> \
-n trident-protect
```

Permissões de IAM de armazenamento compatível com S3

Ao acessar armazenamento compatível com S3, como Amazon S3, S3 genérico, "StorageGRID S3" , ou "ONTAP S3" Ao usar o Trident Protect, você precisa garantir que as credenciais de usuário fornecidas tenham as permissões necessárias para acessar o bucket. Segue abaixo um exemplo de uma política que concede as permissões mínimas necessárias para acesso ao Trident Protect. Você pode aplicar essa política ao usuário que gerencia as políticas de bucket compatíveis com o S3.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:GetObject",
        "s3:ListBucket",
        "s3:DeleteObject"
      ],
      "Resource": "*"
```

```
    }
  ]
}
```

Para obter mais informações sobre as políticas do Amazon S3, consulte os exemplos no "Documentação do Amazon S3" .

Identidade do Pod EKS para autenticação do Amazon S3 (AWS)

O Trident Protect oferece suporte à identidade do EKS Pod para operações de movimentação de dados do Kopia. Este recurso permite acesso seguro aos buckets do S3 sem armazenar credenciais da AWS em segredos do Kubernetes.

Requisitos para a identidade do EKS Pod com o Trident Protect

Antes de usar o EKS Pod Identity com o Trident Protect, certifique-se do seguinte:

- Seu cluster EKS tem a Identidade do Pod habilitada.
- Você criou uma função do IAM com as permissões de bucket do S3 necessárias. Para saber mais, consulte"Permissões de IAM de armazenamento compatível com S3" .
- A função IAM está associada às seguintes contas de serviço do Trident Protect:
  - `<trident-protect>-controller-manager`
  - `<trident-protect>-resource-backup`
  - `<trident-protect>-resource-restore`
  - `<trident-protect>-resource-delete`

Para obter instruções detalhadas sobre como habilitar a identidade do pod e associar funções do IAM a contas de serviço, consulte o "Documentação de identidade do pod AWS EKS" .

Configuração do AppVault Ao usar o EKS Pod Identity, configure seu AppVault CR com o `useIAM: true` sinalizador em vez de credenciais explícitas:

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: eks-protect-vault
  namespace: trident-protect
spec:
  providerType: AWS
  providerConfig:
    s3:
      bucketName: trident-protect-aws
      endpoint: s3.example.com
      useIAM: true
```

Exemplos de geração de chaves AppVault para provedores de nuvem

Ao definir um AppVault CR, você precisa incluir credenciais para acessar os recursos hospedados pelo provedor, a menos que esteja usando a autenticação do IAM. A maneira como você gera as chaves para as credenciais varia de acordo com o provedor. A seguir estão exemplos de geração de chaves de linha de comando para vários provedores. Você pode usar os exemplos a seguir para criar chaves para as credenciais de cada provedor de nuvem.

Google Cloud

```
kubectl create secret generic <secret-name> \
--from-file=credentials=<mycreds-file.json> \
-n trident-protect
```

Amazon S3 (AWS)

```
kubectl create secret generic <secret-name> \
--from-literal=accessKeyID=<objectstorage-accesskey> \
--from-literal=secretAccessKey=<amazon-s3-trident-protect-src-bucket
-secret> \
-n trident-protect
```

Microsoft Azure

```
kubectl create secret generic <secret-name> \
--from-literal=accountKey=<secret-name> \
-n trident-protect
```

Genérico S3

```
kubectl create secret generic <secret-name> \
--from-literal=accessKeyID=<objectstorage-accesskey> \
--from-literal=secretAccessKey=<generic-s3-trident-protect-src-bucket
-secret> \
-n trident-protect
```

ONTAP S3

```
kubectl create secret generic <secret-name> \
--from-literal=accessKeyID=<objectstorage-accesskey> \
--from-literal=secretAccessKey=<ontap-s3-trident-protect-src-bucket
-secret> \
-n trident-protect
```

StorageGRID S3

```
kubectl create secret generic <secret-name> \
--from-literal=accessKeyID=<objectstorage-accesskey> \
--from-literal=secretAccessKey=<storagegrid-s3-trident-protect-src
-bucket-secret> \
-n trident-protect
```

Exemplos de criação do AppVault

A seguir estão exemplos de definições do AppVault para cada provedor.

Exemplos do AppVault CR

Você pode usar os exemplos CR a seguir para criar objetos AppVault para cada provedor de nuvem.

OBSERVAÇÃO

- Opcionalmente, você pode especificar um segredo do Kubernetes que contém senhas personalizadas para a criptografia do repositório Restic e Kopia. Senhas do repositório do controlador de dadosConsulte para obter mais informações.
- Para objetos do Amazon S3 (AWS) AppVault, você pode especificar opcionalmente um sessionToken, o que é útil se você estiver usando SSO (logon único) para autenticação. Esse token é criado quando você gera chaves para o provedor no Exemplos de geração de chaves AppVault para provedores de nuvem.
- Para objetos S3 AppVault, você pode opcionalmente especificar um URL de proxy de saída para tráfego S3 de saída usando a `spec.providerConfig.S3.proxyURL` chave.

Google Cloud

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: gcp-trident-protect-src-bucket
  namespace: trident-protect
spec:
  dataMoverPasswordSecretRef: my-optional-data-mover-secret
  providerType: GCP
  providerConfig:
    gcp:
      bucketName: trident-protect-src-bucket
      projectID: project-id
  providerCredentials:
    credentials:
      valueFromSecret:
        key: credentials
```

```
        name: gcp-trident-protect-src-bucket-secret
```

Amazon S3 (AWS)

```
---
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: amazon-s3-trident-protect-src-bucket
  namespace: trident-protect
spec:
  dataMoverPasswordSecretRef: my-optional-data-mover-secret
  providerType: AWS
  providerConfig:
    s3:
      bucketName: trident-protect-src-bucket
      endpoint: s3.example.com
      proxyURL: http://10.1.1.1:3128
  providerCredentials:
    accessKeyID:
      valueFromSecret:
        key: accessKeyID
        name: s3-secret
    secretAccessKey:
      valueFromSecret:
        key: secretAccessKey
        name: s3-secret
    sessionToken:
      valueFromSecret:
        key: sessionToken
        name: s3-secret
```

OBSERVAÇÃO Para ambientes EKS usando Pod Identity com o movimentador de dados Kopia, você pode remover o `providerCredentials` seção e adicionar `useIAM: true` sob o `s3` configuração em vez disso.

Microsoft Azure

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: azure-trident-protect-src-bucket
  namespace: trident-protect
spec:
  dataMoverPasswordSecretRef: my-optional-data-mover-secret
  providerType: Azure
```

```
  providerConfig:
    azure:
      accountName: account-name
      bucketName: trident-protect-src-bucket
  providerCredentials:
    accountKey:
      valueFromSecret:
        key: accountKey
        name: azure-trident-protect-src-bucket-secret
```

Genérico S3

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: generic-s3-trident-protect-src-bucket
  namespace: trident-protect
spec:
  dataMoverPasswordSecretRef: my-optional-data-mover-secret
  providerType: GenericS3
  providerConfig:
    s3:
      bucketName: trident-protect-src-bucket
      endpoint: s3.example.com
      proxyURL: http://10.1.1.1:3128
  providerCredentials:
    accessKeyID:
      valueFromSecret:
        key: accessKeyID
        name: s3-secret
    secretAccessKey:
      valueFromSecret:
        key: secretAccessKey
        name: s3-secret
```

ONTAP S3

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: ontap-s3-trident-protect-src-bucket
  namespace: trident-protect
spec:
  dataMoverPasswordSecretRef: my-optional-data-mover-secret
  providerType: OntapS3
```

```
  providerConfig:
    s3:
      bucketName: trident-protect-src-bucket
      endpoint: s3.example.com
      proxyURL: http://10.1.1.1:3128
  providerCredentials:
    accessKeyID:
      valueFromSecret:
        key: accessKeyID
        name: s3-secret
    secretAccessKey:
      valueFromSecret:
        key: secretAccessKey
        name: s3-secret
```

StorageGRID S3

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: storagegrid-s3-trident-protect-src-bucket
  namespace: trident-protect
spec:
  dataMoverPasswordSecretRef: my-optional-data-mover-secret
  providerType: StorageGridS3
  providerConfig:
    s3:
      bucketName: trident-protect-src-bucket
      endpoint: s3.example.com
      proxyURL: http://10.1.1.1:3128
  providerCredentials:
    accessKeyID:
      valueFromSecret:
        key: accessKeyID
        name: s3-secret
    secretAccessKey:
      valueFromSecret:
        key: secretAccessKey
        name: s3-secret
```

Exemplos de criação do AppVault usando a CLI do Trident Protect

Você pode usar os seguintes exemplos de comandos CLI para criar o AppVault CRS para cada provedor.

OBSERVAÇÃO

- Opcionalmente, você pode especificar um segredo do Kubernetes que contém

senhas personalizadas para a criptografia do repositório Restic e Kopia. Senhas do repositório do controlador de dadosConsulte para obter mais informações.

- Para objetos S3 AppVault, você pode opcionalmente especificar um URL de proxy de saída para tráfego S3 de saída usando o `--proxy-url <ip_address:port>` argumento.

Google Cloud

```
tridentctl-protect create vault GCP <vault-name> \
--bucket <mybucket> \
--project <my-gcp-project> \
--secret <secret-name>/credentials \
--data-mover-password-secret-ref <my-optional-data-mover-secret> \
-n trident-protect
```

Amazon S3 (AWS)

```
tridentctl-protect create vault AWS <vault-name> \
--bucket <bucket-name> \
--secret  <secret-name>  \
--endpoint <s3-endpoint> \
--data-mover-password-secret-ref <my-optional-data-mover-secret> \
-n trident-protect
```

Microsoft Azure

```
tridentctl-protect create vault Azure <vault-name> \
--account <account-name> \
--bucket <bucket-name> \
--secret <secret-name> \
--data-mover-password-secret-ref <my-optional-data-mover-secret> \
-n trident-protect
```

Genérico S3

```
tridentctl-protect create vault GenericS3 <vault-name> \
--bucket <bucket-name> \
--secret  <secret-name>  \
--endpoint <s3-endpoint> \
--data-mover-password-secret-ref <my-optional-data-mover-secret> \
-n trident-protect
```

ONTAP S3

```
tridentctl-protect create vault OntapS3 <vault-name> \
--bucket <bucket-name> \
--secret  <secret-name>  \
--endpoint <s3-endpoint> \
--data-mover-password-secret-ref <my-optional-data-mover-secret> \
-n trident-protect
```

StorageGRID S3

```
tridentctl-protect create vault StorageGridS3 <vault-name> \
--bucket <bucket-name> \
--secret  <secret-name>  \
--endpoint <s3-endpoint> \
--data-mover-password-secret-ref <my-optional-data-mover-secret> \
-n trident-protect
```

Apoiado `providerConfig.s3` opções de configuração

Consulte a tabela a seguir para obter as opções de configuração do provedor S3:

Parâmetro	Descrição	Padrão	Exemplo
`providerConfig.s3.skipCertValidation`	Desative a verificação de certificado SSL/TLS.	falso	"verdadeiro", "falso"
`providerConfig.s3.secure`	Habilite a comunicação HTTPS segura com o endpoint S3.	verdadeiro	"verdadeiro", "falso"
`providerConfig.s3.proxyURL`	Especifique o URL do servidor proxy usado para conectar-se ao S3.	Nenhum	http://proxy.example.com:8080
`providerConfig.s3.rootCA`	Forneça um certificado CA raiz personalizado para verificação SSL/TLS.	Nenhum	"CN=MyCustomCA"
`providerConfig.s3.useIAM`	Habilite a autenticação IAM para acessar os buckets do S3. Aplicável à identidade do Pod EKS.	falso	verdadeiro, falso

Ver informações do AppVault

Você pode usar o plugin Trident Protect CLI para visualizar informações sobre os objetos do AppVault que você criou no cluster.

Passos

1. Exibir o conteúdo de um objeto AppVault:

```
tridentctl-protect get appvaultcontent gcp-vault \
--show-resources all \
-n trident-protect
```

Exemplo de saída:

```
+-------------+-------+----------+----------------------------
+--------------------------+
|   CLUSTER   |  APP  |   TYPE   |            NAME             |
TIMESTAMP         |
+-------------+-------+----------+----------------------------
+--------------------------+
|             | mysql | snapshot | mysnap                      | 2024-
08-09 21:02:11 (UTC) |
| production1 | mysql | snapshot | hourly-e7db6-20240815180300 | 2024-
08-15 18:03:06 (UTC) |
| production1 | mysql | snapshot | hourly-e7db6-20240815190300 | 2024-
08-15 19:03:06 (UTC) |
| production1 | mysql | snapshot | hourly-e7db6-20240815200300 | 2024-
08-15 20:03:06 (UTC) |
| production1 | mysql | backup   | hourly-e7db6-20240815180300 | 2024-
08-15 18:04:25 (UTC) |
| production1 | mysql | backup   | hourly-e7db6-20240815190300 | 2024-
08-15 19:03:30 (UTC) |
| production1 | mysql | backup   | hourly-e7db6-20240815200300 | 2024-
08-15 20:04:21 (UTC) |
| production1 | mysql | backup   | mybackup5                   | 2024-
08-09 22:25:13 (UTC) |
|             | mysql | backup   | mybackup                    | 2024-
08-09 21:02:52 (UTC) |
+-------------+-------+----------+----------------------------
+--------------------------+
```

2. Opcionalmente, para ver o AppVaultPath para cada recurso, use o `--show-paths` sinalizador .

   O nome do cluster na primeira coluna da tabela só estará disponível se um nome de cluster tiver sido especificado na instalação do Trident Protect no Helm. Por exemplo: `--set clusterName=production1` .

Remova um AppVault

Você pode remover um objeto AppVault a qualquer momento.

OBSERVAÇÃO	Não remova a `finalizers` chave no AppVault CR antes de excluir o objeto AppVault. Se você fizer isso, isso pode resultar em dados residuais no bucket do AppVault e recursos órfãos no cluster.

Antes de começar

Certifique-se de que você excluiu todos os CRS de snapshot e backup que estão sendo usados pelo AppVault que deseja excluir.

Remova um AppVault usando a CLI do Kubernetes

1. Remova o objeto AppVault, substituindo `appvault-name` pelo nome do objeto AppVault para remover:

```
kubectl delete appvault <appvault-name> \
-n trident-protect
```

Remova um AppVault usando a CLI do Trident Protect.

1. Remova o objeto AppVault, substituindo `appvault-name` pelo nome do objeto AppVault para remover:

```
tridentctl-protect delete appvault <appvault-name> \
-n trident-protect
```

Criptografe os metadados do AppVault para snapshots e backups do Trident Protect

Você pode criptografar os metadados de snapshot e backup que o Trident Protect armazena em um AppVault, incluindo informações confidenciais em Kubernetes Secrets. Você também pode alterar a senha que protege os dados criptografados. Uma única alteração de senha criptografa novamente a chave de criptografia de metadados e move cada repositório Kopia e cada repositório Restic no AppVault para a nova senha.

A criptografia está desativada por padrão. Ao ativá-la, o Trident Protect criptografa os novos metadados de snapshot e backup antes de armazenar os metadados no AppVault. Isso protege os recursos do seu aplicativo, incluindo informações confidenciais em Secrets, a menos que você exclua explicitamente esses recursos do snapshot ou backup do aplicativo.

Uma senha do cofre protege a chave que criptografa os dados. Observe o seguinte comportamento:

- Ao alterar a senha do cofre, o Trident Protect protege a mesma chave com a nova senha. Ele não sobrescreve os dados que já estão criptografados.
- Os dados armazenados antes de você ativar a criptografia permanecem legíveis.
- Os dados criptografados permanecem legíveis, mesmo se você desativar a criptografia para novos dados posteriormente.

Sobre esta tarefa

Ao ativar a criptografia, o Trident Protect usa uma senha padrão insegura, a menos que você especifique uma senha personalizada. Para cada AppVault, crie uma senha personalizada e faça referência a ela em `spec.vaultPasswordSecretRef`.

Se você habilitou a criptografia com a senha padrão e adicionou uma senha personalizada depois, deve fazer a rotação da senha para que os dados e repositórios existentes passem para a senha personalizada. Consulte Gire a senha do cofre.

Use uma senha personalizada exclusiva para cada AppVault. Altere a senha de um AppVault por vez e confirme que a rotação foi concluída antes de iniciar outra alteração de senha.

Antes de começar

Certifique-se de ter o seguinte:

- Acesso ao `trident-protect` namespace.
- Permissão para criar Secrets do Kubernetes e aplicar patches em recursos personalizados (CRs) do AppVault.
- Um Secret de senha separado está planejado para cada AppVault.

Ativar a criptografia de metadados

Você habilita a criptografia de metadados definindo um valor do Helm para o Trident Protect.

Passos

1. Defina `resourceBackupEncryptionEnabled` como `true` usando um `values.yaml` arquivo:
   a. Adicione o seguinte ao seu `values.yaml` arquivo:

   ```
   resourceBackupEncryptionEnabled: true
   ```

   b. Aplique o arquivo:

   ```
   helm upgrade trident-protect -n trident-protect netapp-trident-protect/trident-protect -f values.yaml --reuse-values
   ```

2. Alternativamente, defina `resourceBackupEncryptionEnabled` como `true` usando a flag `--set` do Helm:

   ```
   helm upgrade trident-protect -n trident-protect netapp-trident-protect/trident-protect \
     --set resourceBackupEncryptionEnabled=true \
     --reuse-values
   ```

Resultado

O Trident Protect criptografa os metadados de novos snapshots e backups.

O que se segue?

Crie uma senha personalizada para cada AppVault. Até que você faça isso, o Trident Protect usa a senha padrão insegura.

Crie uma senha personalizada para o cofre

Para usar sua própria senha, crie um segredo do Kubernetes que contenha a senha e, em seguida, faça referência ao segredo no AppVault.

Sobre esta tarefa

Se você configurar a senha personalizada ao criar o AppVault, antes que qualquer dado seja armazenado, o AppVault usará a senha personalizada imediatamente. Você não precisa fazer a rotação da senha.

Se o AppVault já tiver repositórios Kopia ou Restic, adicionar uma nova senha não altera a senha desses repositórios. Não use as etapas desta seção. Em vez disso, siga as etapas em Gire a senha do cofre.

O `VAULT_PASSWORD` valor deve atender aos seguintes requisitos:

- Não deve estar vazio
- Deve ser menor que 64 KiB
- Não deve conter um caractere NUL
- Não deve conter quebra de linha
- Não deve começar nem terminar com um espaço ou outro caractere em branco

A senha pode conter outros caracteres, incluindo caracteres não ASCII.

Passos

1. Crie um Secret no mesmo namespace que o AppVault. O Secret deve conter uma chave chamada `VAULT_PASSWORD`:

```
kubectl -n trident-protect create secret generic vault-password \
  --from-literal=VAULT_PASSWORD='<password>'
```

OBSERVAÇÃO	Se o AppVault informar que a senha não é válida, verifique o `VAULT_PASSWORD` valor em relação aos requisitos acima. Causas comuns são um valor vazio, uma quebra de linha no final ou espaços no início ou no final da senha.

2. Consulte o Secret no campo `spec.vaultPasswordSecretRef`do CR AppVault:

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: my-vault
  namespace: trident-protect
spec:
  vaultPasswordSecretRef: vault-password
  providerType: <provider-type>
  providerConfig:
```

```
    <provider-configuration>
```

Se `spec.vaultPasswordSecretRef` estiver vazio, o AppVault usa a senha padrão insegura.

Resultado

O AppVault usa sua senha personalizada para proteger a chave de criptografia de metadados.

Gire a senha do cofre

Você pode rotacionar a senha de um AppVault que já armazena dados. Uma rotação criptografa novamente a chave de criptografia de metadados e move cada repositório Kopia e cada repositório Restic no AppVault para a nova senha.

Sobre esta tarefa

Durante a rotação, o Trident Protect faz o seguinte:

- Aguarda a conclusão das tarefas de backup e restauração em execução do AppVault antes de iniciar.
- Protege a chave de criptografia de metadados (`metadata-key.json`) com a nova senha. Não sobrescreve os objetos de arquivamento.
- Localiza cada repositório Kopia e cada repositório Restic sob o prefixo AppVault e altera sua senha para a nova senha. Os repositórios que já abrem com a nova senha não são alterados.
- Ignora os repositórios que um trabalho de backup ou restauração ainda está usando e tenta novamente mais tarde.
- Suspende os agendamentos habilitados que fazem referência ao AppVault. Para cada agendamento, o Trident Protect define `spec.enabled` como `false` e adiciona a seguinte anotação:

```
protect.trident.netapp.io/vault-password-rotation-suspended:
<namespace>/<appvault-name>
```

Quando a rotação termina, o Trident Protect reativa apenas os agendamentos que possuem essa anotação. Se uma rotação falhar, ela continua de onde parou. Os repositórios que já utilizam a nova senha não são alterados novamente.

Antes de começar

Certifique-se de ter o seguinte:

- Uma nova senha que atenda aos requisitos de senha do cofre.
- Nenhum outro AppVault está rotacionando sua senha. Se vários AppVaults compartilharem o mesmo bucket e prefixo, apenas um deles poderá rotacionar a senha. Consulte Gerencie a rotação para os AppVaults que compartilham um bucket.

Passos

1. Crie um novo Secret com um nome diferente do nome do Secret atual:

```
kubectl -n trident-protect create secret generic vault-password-new \
  --from-literal=VAULT_PASSWORD='<new-password>'
```

IMPORTANTE	Não edite nem exclua o Secret atual antes que a rotação termine. O Trident Protect precisa da senha atual para abrir os dados existentes até que todos os repositórios usem a nova senha. Se você excluir o Secret atual cedo demais, recrie-o com o mesmo nome e o valor `VAULT_PASSWORD`original.

2. Ative a rotação e configure o AppVault para usar o novo Secret. Substitua `my-vault` pelo nome do seu AppVault:

```
kubectl -n trident-protect patch appvault my-vault --type merge --patch
'
spec:
  enableRotation: true
  vaultPasswordSecretRef: vault-password-new
'
```

OBSERVAÇÃO	Você deve definir `spec.enableRotation``true` `true` antes de alterar a senha. Se estiver como `false` `false`, o AppVault informa que a rotação está desativada e não altera a senha.

3. Monitore a rotação seguindo os passos em Monitore a rotação de senhas.
4. Após o término da rotação, exclua o Secret anterior caso nada mais o utilize.

Resultado

A chave de criptografia dos metadados e todos os repositórios do Kopia e do Restic no AppVault estão protegidos pela nova senha, e os agendamentos que o Trident Protect havia suspendido estão habilitados novamente.

Monitore a rotação de senhas

Você pode verificar o progresso da rotação de senhas e confirmar que ela está concluída.

Passos

1. Veja o AppVault:

```
kubectl -n trident-protect get appvault my-vault --output yaml
```

2. Na saída, verifique a `status.vaultPasswordRotation` seção. Enquanto a rotação estiver em andamento, ela terá a seguinte aparência:

```
status:
  vaultPasswordRotation:
    state: Rotating
    rotated: 3
    pending: 1
    message: <message>
```

Esta tabela descreve os campos:

Campo	Descrição
status.vaultPasswordRotation.state	Estado de rotação atual.
status.vaultPasswordRotation.rotated	Número de repositórios que agora usam a nova senha.
status.vaultPasswordRotation.pending	Número de repositórios que foram ignorados porque estão sendo usados em um backup ou restauração.
status.vaultPasswordRotation.message	Última mensagem de progresso ou falha.

Resultado

A rotação estará concluída quando todas as condições a seguir forem verdadeiras:

- `status.vaultPasswordRotation` não está presente.
- `status.previousRepositoryPasswordSecretRef` está vazio.
- `status.repositoryPasswordSecretRef` correspondências `spec.vaultPasswordSecretRef`.

O que se segue?

Se a rotação não terminar, verifique o seguinte:

- **A rotação não inicia**: `spec.enableRotation` pode ser `false`. Defina-a como `true`, e então defina `spec.vaultPasswordSecretRef` para o novo Secret.
- **Repositórios permanecem pendentes**: tarefas de backup ou restauração ainda estão usando esses repositórios. Aguarde a conclusão das tarefas. O Trident Protect tenta novamente os repositórios pendentes automaticamente.
- **A rotação não inicia porque outro AppVault possui o bloqueio**: consulte Gerencie a rotação para os AppVaults que compartilham um bucket.
- **O campo de mensagem reporta uma falha**: resolva o problema descrito na mensagem. A rotação será retomada de onde parou.

Migrar de dataMoverPasswordSecretRef

Se um AppVault existente usa o campo `spec.dataMoverPasswordSecretRef` antigo, você pode movê-lo para `spec.vaultPasswordSecretRef`.

Sobre esta tarefa

O Trident Protect mantém `spec.dataMoverPasswordSecretRef` para compatibilidade com versões anteriores. Use `spec.vaultPasswordSecretRef` para novos AppVaults.

Passos

1. Mantenha `spec.dataMoverPasswordSecretRef` e seu Secret no lugar.
2. Altere a senha para um novo Secret usando `spec.vaultPasswordSecretRef`. Siga as etapas em Gire a senha do cofre.
3. Após a rotação terminar, remova `spec.dataMoverPasswordSecretRef` do AppVault.

4. Exclua o Secret antigo se nada mais o estiver usando.

Gerencie a rotação para os AppVaults que compartilham um bucket

Se vários AppVaults usarem o mesmo bucket e prefixo, apenas um deles poderá rotacionar a senha.

Sobre esta tarefa

A rotação de senhas usa um arquivo de bloqueio chamado `password-rotation.lock`. O arquivo de bloqueio registra o UID do AppVault autorizado a rotacionar a senha do bucket e do prefixo. Se um segundo AppVault tiver `spec.enableRotation: true` e o arquivo de bloqueio nomear um AppVault diferente, o segundo AppVault não poderá fazer a rotação.

Exemplo 1. Passos

Transferir a rotação para outro AppVault

1. No AppVault que possui o bloqueio, limpe `spec.enableRotation`:

```
kubectl -n trident-protect patch appvault <current-owner> --type
merge --patch '
spec:
  enableRotation: false
'
```

2. Aguarde até que o bloqueio seja removido.
3. Na AppVault que deve ser rotacionada em seguida, defina `spec.enableRotation` como `true`:

```
kubectl -n trident-protect patch appvault <new-owner> --type merge
--patch '
spec:
  enableRotation: true
'
```

Remova um bloqueio deixado por um AppVault excluído

Use estas etapas se o AppVault que possui o bloqueio não existir mais e o bloqueio não tiver sido removido, por exemplo, após uma falha.

1. Exclua o `password-rotation.lock` arquivo do prefixo AppVault no seu bucket.
2. No AppVault que deve ser rotacionado, defina `spec.enableRotation` como `true`:

```
kubectl -n trident-protect patch appvault <appvault-name> --type
merge --patch '
spec:
  enableRotation: true
'
```

Informações relacionadas

- "Use objetos do Trident Protect AppVault para gerenciar buckets."
- "Personalize a instalação do Trident Protect"
- "Proteger aplicações"
- "Restaure aplicativos usando o Trident Protect."

Defina uma aplicação para gestão com o Trident Protect.

Você pode definir um aplicativo que deseja gerenciar com o Trident Protect criando um CR (Create Request) de aplicativo e um CR do AppVault associado.

Crie um AppVault CR

Você precisa criar um AppVault CR que será usado ao executar operações de proteção de dados no aplicativo, e o AppVault CR precisa residir no cluster onde o Trident Protect está instalado. O pedido de configuração (CR) do AppVault é específico para o seu ambiente; para exemplos de pedidos de configuração do AppVault, consulte [link para a documentação]."Recursos personalizados do AppVault."

Definir uma aplicação

Você precisa definir cada aplicativo que deseja gerenciar com o Trident Protect. Você pode definir um aplicativo para gerenciamento criando manualmente uma solicitação de configuração (CR) do aplicativo ou usando a CLI do Trident Protect.

OBSERVAÇÃO	Para definir um aplicativo com escopo de VM que protege máquinas virtuais (VMs) específicas do KubeVirt por nome em vez de namespaces inteiros, use o campo `spec.includedVirtualMachines` no CR do aplicativo ou o sinalizador `--virtual-machines` na CLI. Para obter detalhes, referência de campos e exemplos de YAML, consulte "Proteja máquinas virtuais KubeVirt com Trident Protect".

Adicione uma aplicação utilizando um CR

Passos

1. Criar o ficheiro CR da aplicação de destino:
   a. Crie o arquivo de recurso personalizado (CR) e nomeie-o (por exemplo, `maria-app.yaml`).
   b. Configure os seguintes atributos:
      - **metadata.name**: (*required*) o nome do recurso personalizado do aplicativo. Observe o nome escolhido porque outros arquivos CR necessários para operações de proteção referem-se a esse valor.
      - **spec.includedNamespaces**: (*required*) Use o seletor de namespace e rótulo para especificar os namespaces e recursos que o aplicativo usa. O namespace do aplicativo deve fazer parte dessa lista. O seletor de etiquetas é opcional e pode ser usado para filtrar recursos dentro de cada namespace especificado.
      - **spec.includedClusterScopedResources**: (*Optional*) Use este atributo para especificar recursos com escopo de cluster a serem incluídos na definição do aplicativo. Esse atributo permite que você selecione esses recursos com base em seu grupo, versão, tipo e rótulos.

        - **GroupVersionKind**: (*required*) especifica o grupo API, a versão e o tipo do recurso com escopo de cluster.
        - **LabelSeletor**: (*Opcional*) filtra os recursos com escopo de cluster com base em seus rótulos.
    - **spec.volumeGroups**: (*Opcional*) use este atributo para criar snapshots de grupo de volume consistentes com falhas de volumes relacionados em conjunto. Para obter detalhes, consulte "Use snapshots de grupo de volume com Trident Protect".
    - **metadata.annotations.protect.trident.netapp.io/skip-vm-freeze**: (*Opcional*) Esta anotação só se aplica a aplicações definidas a partir de máquinas virtuais, como em ambientes KubeVirt, onde o congelamento do sistema de arquivos ocorre antes dos snapshots. Especifique se este aplicativo pode gravar no sistema de arquivos durante um snapshot. Se definido como verdadeiro, o aplicativo ignora a configuração global e pode gravar no sistema de arquivos durante um snapshot. Se definido como falso, o aplicativo ignora a configuração global e o sistema de arquivos é congelado durante a captura de um instantâneo. Se especificada, mas a aplicação não tiver máquinas virtuais na definição da aplicação, a anotação será ignorada. Caso não seja especificado, o aplicativo segue o"configuração de congelamento global do Trident Protect" .

Se você precisar aplicar essa anotação depois que um aplicativo já tiver sido criado, você pode usar o seguinte comando:

```
kubectl annotate application -n <application CR namespace> <application CR
name> protect.trident.netapp.io/skip-vm-freeze="true"
```

+
Exemplo YAML:

+

```
apiVersion: protect.trident.netapp.io/v1
kind: Application
metadata:
  annotations:
    protect.trident.netapp.io/skip-vm-freeze: "false"
  name: my-app-name
  namespace: my-app-namespace
spec:
  includedNamespaces:
    - namespace: namespace-1
      labelSelector:
        matchLabels:
          app: example-app
    - namespace: namespace-2
      labelSelector:
        matchLabels:
```

```
        app: another-example-app
includedClusterScopedResources:
  - groupVersionKind:
      group: rbac.authorization.k8s.io
      kind: ClusterRole
      version: v1
    labelSelector:
      matchLabels:
        mylabel: test
```

1. (*Opcional*) Se necessário, você pode adicionar filtragem de recursos à mesma CR para incluir ou excluir recursos específicos:

   - **Exemplo de filtro genérico:**

     - **ResourceFilter.resourceSelectionCriteria**: (Necessário para filtragem) Use `Include` ou `Exclude` inclua ou exclua um recurso definido em resourceMatchers. Adicione os seguintes parâmetros resourceMatchers para definir os recursos a serem incluídos ou excluídos:

       - **ResourceFilter.resourceMatchers**: Uma matriz de resourceMatcher objetos. Se você definir vários elementos nesse array, eles corresponderão como uma OPERAÇÃO OU, e os campos dentro de cada elemento (grupo, tipo, versão) corresponderão como uma OPERAÇÃO E.
       - **ResourceMatchers[].group**: (*Optional*) Grupo do recurso a ser filtrado.
       - **ResourceMatchers[].kind**: (*Opcional*) tipo do recurso a ser filtrado.
       - **ResourceMatchers[].version**: (*Optional*) versão do recurso a ser filtrado.
       - **ResourceMatchers[].names**: (*Optional*) nomes no campo Kubernetes metadata.name do recurso a ser filtrado.
       - **ResourceMatchers[].namespaces**: (*Optional*) namespaces no campo Kubernetes metadata.name do recurso a ser filtrado.
       - **ResourceMatchers[].labelSelectors**: (*Optional*) string de seleção de etiquetas no campo Kubernetes metadata.name do recurso, conforme definido no "Documentação do Kubernetes". Por exemplo `"trident.netapp.io/os=linux"`: .

OBSERVAÇÃO	Quando ambos `resourceFilter` e `labelSelector` são usados, `resourceFilter` corre primeiro e depois `labelSelector` é aplicado aos recursos resultantes.

     Por exemplo:

```
spec:
  resourceFilter:
    resourceSelectionCriteria: "Include"
    resourceMatchers:
      - group: my-resource-group-1
        kind: my-resource-kind-1
```

```
        version: my-resource-version-1
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
      - group: my-resource-group-2
        kind: my-resource-kind-2
        version: my-resource-version-2
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
```

- **Exemplo de filtro somente de PVC:**

  Para definir um aplicativo somente para PVC, você também deve incluir `PersistentVolume` e `VolumeSnapshotClass` no filtro de recursos. As operações de Snapshot e backup dependem de `PersistentVolume` (o volume com escopo de cluster vinculado a cada PVC) e `VolumeSnapshotClass` (o driver de snapshot), e falharão sem eles. Por exemplo:

```
apiVersion: protect.trident.netapp.io/v1
kind: Application
metadata:
  name: my-pvc-app
  namespace: my-app-namespace
spec:
  includedNamespaces:
  - namespace: my-app-namespace
  resourceFilter:
    resourceMatchers:
    - kind: PersistentVolumeClaim
      version: v1
    - kind: PersistentVolume
      version: v1
    - kind: VolumeSnapshotClass
      version: v1
    resourceSelectionCriteria: Include
```

2. Depois de criar a aplicação CR para corresponder ao seu ambiente, aplique o CR. Por exemplo:

```
kubectl apply -f maria-app.yaml
```

Passos

1. Crie e aplique a definição do aplicativo usando um dos exemplos a seguir, substituindo valores entre parênteses por informações do ambiente. Você pode incluir namespaces e recursos na definição do aplicativo usando listas separadas por vírgulas com os argumentos mostrados nos exemplos.

Você pode, opcionalmente, usar uma anotação ao criar um aplicativo para especificar se o aplicativo pode gravar no sistema de arquivos durante um snapshot. Isso só se aplica a aplicativos definidos a partir de máquinas virtuais, como em ambientes KubeVirt, onde o congelamento do sistema de arquivos ocorre antes dos snapshots. Se você definir a anotação para `true` , o aplicativo ignora a configuração global e pode gravar no sistema de arquivos durante um snapshot. Se você definir para `false` , o aplicativo ignora a configuração global e o sistema de arquivos fica congelado durante a captura de um snapshot. Se você usar a anotação, mas o aplicativo não tiver máquinas virtuais na definição do aplicativo, a anotação será ignorada. Se você não usar a anotação, o aplicativo seguirá o padrão."configuração de congelamento global do Trident Protect" .

Para especificar a anotação ao usar a CLI para criar um aplicativo, você pode usar a `--annotation` flag.

+ * Crie o aplicativo e utilize a configuração global para o comportamento de congelamento do sistema de arquivos:

+

```
tridentctl-protect create application <my_new_app_cr_name> --namespaces
<namespaces_to_include> --csr <cluster_scoped_resources_to_include>
--namespace <my-app-namespace>
```

+ * Crie o aplicativo e configure as definições locais do aplicativo para o comportamento de congelamento do sistema de arquivos:

+

```
tridentctl-protect create application <my_new_app_cr_name> --namespaces
<namespaces_to_include> --csr <cluster_scoped_resources_to_include>
--namespace <my-app-namespace> --annotation
protect.trident.netapp.io/skip-vm-freeze=<"true"|"false">
```

Você pode usar `--resource-filter-include` e `--resource-filter-exclude` sinalizadores para incluir ou excluir recursos com base em `resourceSelectionCriteria` como grupo, tipo, versão, rótulos, nomes e namespaces, conforme mostrado no exemplo a seguir:

+

```
tridentctl-protect create application <my_new_app_cr_name> --namespaces
<namespaces_to_include> --csr <cluster_scoped_resources_to_include>
--namespace <my-app-namespace> --resource-filter-include
'[{"Group":"apps","Kind":"Deployment","Version":"v1","Names":["my-
deployment"],"Namespaces":["my-namespace"],"LabelSelectors":["app=my-
app"]}]'
```

+ * Para definir um aplicativo somente para PVC, você também deve incluir `PersistentVolume` e `VolumeSnapshotClass` no filtro de recursos. As operações de Snapshot e backup dependem de

`PersistentVolume` (o volume com escopo de cluster vinculado a cada PVC) e `VolumeSnapshotClass` (o driver de Snapshot), e falharão sem eles. Por exemplo:

+

```
tridentctl-protect create app my-pvc-app --namespaces <my-app-namespace> --resource-filter-include '[{"Kind":"PersistentVolumeClaim","Version":"v1"},{"Kind":"PersistentVolume","Version":"v1"},{"Kind":"VolumeSnapshotClass","Version":"v1"}]' -n <my-app-namespace>
```

Proteja aplicativos usando o Trident Protect.

Você pode proteger todos os aplicativos gerenciados pelo Trident Protect criando snapshots e backups usando uma política de proteção automatizada ou de forma pontual.

OBSERVAÇÃO

- Você pode configurar o Trident Protect para congelar e descongelar sistemas de arquivos durante operações de proteção de dados. "Saiba mais sobre como configurar o congelamento do sistema de arquivos com o Trident Protect.".
- Para capturar instantâneos dos volumes relacionados de um aplicativo no mesmo ponto no tempo, defina grupos de volume no aplicativo. Consulte "Use snapshots de grupo de volume com Trident Protect".

Crie um snapshot sob demanda

Você pode criar um snapshot sob demanda a qualquer momento.

OBSERVAÇÃO

Os recursos com escopo de cluster são incluídos em um backup, snapshot ou clone se forem explicitamente referenciados na definição do aplicativo ou se tiverem referências a qualquer um dos namespaces do aplicativo.

Crie um instantâneo usando um CR

Passos

1. Crie o arquivo de recurso personalizado (CR) e nomeie-o `trident-protect-snapshot-cr.yaml`.
2. No arquivo criado, configure os seguintes atributos:
   - **metadata.name**: (*required*) o nome deste recurso personalizado; escolha um nome único e sensível para o seu ambiente.
   - **Spec.applicationRef**: O nome do Kubernetes da aplicação para snapshot.
   - **Spec.appVaultRef**: (*required*) o nome do AppVault onde o conteúdo instantâneo (metadados) deve ser armazenado.
   - **Spec.reclaimPolicy**: (*Optional*) define o que acontece com o AppArchive de um snapshot

quando o snapshot CR é excluído. Isso significa que, mesmo quando definido como `Retain`, o instantâneo será excluído. Opções válidas:

- `Retain` (predefinição)
- `Delete`

```
apiVersion: protect.trident.netapp.io/v1
kind: Snapshot
metadata:
  namespace: my-app-namespace
  name: my-cr-name
spec:
  applicationRef: my-application
  appVaultRef: appvault-name
  reclaimPolicy: Delete
```

3. Depois de preencher o `trident-protect-snapshot-cr.yaml` ficheiro com os valores corretos, aplique o CR:

```
kubectl apply -f trident-protect-snapshot-cr.yaml
```

Crie um instantâneo usando a CLI

Passos

1. Crie o snapshot, substituindo valores entre parênteses por informações do seu ambiente. Por exemplo:

```
tridentctl-protect create snapshot <my_snapshot_name> --appvault
<my_appvault_name> --app <name_of_app_to_snapshot> -n
<application_namespace>
```

Crie um backup sob demanda

Você pode fazer backup de um aplicativo a qualquer momento.

OBSERVAÇÃO	Os recursos com escopo de cluster são incluídos em um backup, snapshot ou clone se forem explicitamente referenciados na definição do aplicativo ou se tiverem referências a qualquer um dos namespaces do aplicativo.

Antes de começar

Certifique-se de que a expiração do token de sessão da AWS seja suficiente para quaisquer operações de backup S3 de longa execução. Se o token expirar durante a operação de backup, a operação pode falhar.

- Consulte a "Documentação da API da AWS" para obter mais informações sobre como verificar a expiração do token de sessão atual.

- Consulte o "Documentação do AWS IAM" para obter mais informações sobre credenciais com recursos da AWS.

Crie uma cópia de segurança utilizando um CR

Passos

1. Crie o arquivo de recurso personalizado (CR) e nomeie-o `trident-protect-backup-cr.yaml`.
2. No arquivo criado, configure os seguintes atributos:
   - **metadata.name**: (*required*) o nome deste recurso personalizado; escolha um nome único e sensível para o seu ambiente.
   - **Spec.applicationRef**: (*required*) o nome do Kubernetes do aplicativo para fazer backup.
   - **Spec.appVaultRef**: (*required*) o nome do AppVault onde o conteúdo de backup deve ser armazenado.
   - **Spec.dataMover**: (*Optional*) Uma cadeia de carateres indicando qual ferramenta de backup usar para a operação de backup. Valores possíveis (sensíveis a maiúsculas e minúsculas):
     - `Restic`
     - `Kopia` (predefinição)
   - **Spec.reclaimPolicy**: (*Optional*) define o que acontece com um backup quando liberado de sua reivindicação. Valores possíveis:
     - `Delete`
     - `Retain` (predefinição)
   - **spec.snapshotRef**: (*Opcional*): Nome do snapshot a ser usado como origem do backup. Se não for fornecido, um instantâneo temporário será criado e feito backup.

     Exemplo YAML:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: Backup
metadata:
  namespace: my-app-namespace
  name: my-cr-name
spec:
  applicationRef: my-application
  appVaultRef: appvault-name
  dataMover: Kopia
```

3. Depois de preencher o `trident-protect-backup-cr.yaml` ficheiro com os valores corretos, aplique o CR:

```
kubectl apply -f trident-protect-backup-cr.yaml
```

Crie um backup usando a CLI

Passos

1. Crie o backup, substituindo valores entre parênteses por informações do seu ambiente. Por exemplo:

```
tridentctl-protect create backup <my_backup_name> --appvault <my-
vault-name> --app <name_of_app_to_back_up> --data-mover
<Kopia_or_Restic> -n <application_namespace>
```

   Opcionalmente, você pode usar o `--full-backup` sinalizador para especificar se um backup deve ser não incremental. Por padrão, todos os backups são incrementais. Quando esse sinalizador é usado, o backup se torna não incremental. É prática recomendada executar um backup completo periodicamente e, em seguida, executar backups incrementais entre backups completos para minimizar o risco associado às restaurações.

Anotações de backup suportadas

A tabela a seguir descreve as anotações que você pode usar ao criar um CR de backup:

Anotação	Tipo	Descrição	Valor padrão
protect.trident.netapp.io/backup completo	cadeia de carateres	Especifica se um backup deve ser não incremental. Definir para `true` Para criar um backup não incremental. A melhor prática é realizar um backup completo periodicamente e, em seguida, realizar backups incrementais entre os backups completos para minimizar o risco associado às restaurações.	"falso"
protect.trident.netapp.io/snapshot-completion-timeout	cadeia de carateres	Tempo máximo permitido para a conclusão da operação de captura instantânea.	"60m"
protect.trident.netapp.io/volume-snapshots-ready-to-use-timeout	cadeia de carateres	Tempo máximo permitido para que os snapshots de volume atinjam o estado pronto para uso.	"30m"
protect.trident.netapp.io/volume-snapshots-created-timeout	cadeia de carateres	Tempo máximo permitido para a criação de snapshots de volume.	"5m"
protect.trident.netapp.io/pvc-bind-timeout-sec	cadeia de carateres	Tempo máximo (em segundos) de espera para que quaisquer PersistentVolumeClaims (PVCs) recém-criados cheguem ao `Bound` fase anterior à falha das operações.	"1200" (20 minutos)

Criar um cronograma de proteção de dados

Uma política de proteção protege um aplicativo criando instantâneos, backups ou ambos em um cronograma definido. Você pode optar por criar snapshots e backups por hora, dia, semana e mês, e pode especificar o número de cópias a serem mantidas. Você pode agendar um backup completo não incremental usando a anotação full-backup-rule. Por padrão, todos os backups são incrementais. Executar um backup completo periodicamente, juntamente com backups incrementais entre eles, ajuda a reduzir o risco associado às restaurações.

OBSERVAÇÃO

- Você pode criar agendamentos apenas para instantâneos definindo `backupRetention` para zero e `snapshotRetention` para um valor maior que zero. Contexto `snapshotRetention` para zero significa que todos os backups agendados ainda criarão instantâneos, mas eles são temporários e serão excluídos imediatamente após a conclusão do backup.
- Os recursos com escopo de cluster são incluídos em um backup, snapshot ou clone se forem explicitamente referenciados na definição do aplicativo ou se tiverem referências a qualquer um dos namespaces do aplicativo.

Crie uma agenda usando um CR

Passos

1. Crie o arquivo de recurso personalizado (CR) e nomeie-o `trident-protect-schedule-cr.yaml`.
2. No arquivo criado, configure os seguintes atributos:
   - **metadata.name**: (*required*) o nome deste recurso personalizado; escolha um nome único e sensível para o seu ambiente.
   - **Spec.dataMover**: (*Optional*) Uma cadeia de carateres indicando qual ferramenta de backup usar para a operação de backup. Valores possíveis (sensíveis a maiúsculas e minúsculas):
     - `Restic`
     - `Kopia` (predefinição)
   - **Spec.applicationRef**: O nome do Kubernetes do aplicativo para fazer backup.
   - **Spec.appVaultRef**: (*required*) o nome do AppVault onde o conteúdo de backup deve ser armazenado.
   - **spec.backupRetention**: (*Obrigatório*) O número de backups a serem retidos. Zero indica que nenhum backup deve ser criado (somente instantâneos).
   - **spec.backupReclaimPolicy**: (Opcional) Determina o que acontece com um backup se o CR de backup for excluído durante seu período de retenção. Após o período de retenção, os backups são sempre apagados. Valores possíveis (diferencia maiúsculas de minúsculas):
     - `Retain` (predefinição)
     - `Delete`
   - **spec.snapshotRetention**: (*Obrigatório*) O número de snapshots a serem retidos. Zero indica que nenhum instantâneo deve ser criado.
   - **spec.snapshotReclaimPolicy**: (Opcional) Determina o que acontece com um snapshot se o CR do snapshot for excluído durante seu período de retenção. Após o período de retenção, os instantâneos são sempre apagados. Valores possíveis (diferencia maiúsculas de minúsculas):
     - `Retain`
     - `Delete`(padrão)
   - **spec.granularity**: a frequência em que o horário deve ser executado. Valores possíveis, juntamente com campos associados obrigatórios:
     - `Hourly`(requer que você especifique `spec.minute`)
     - `Daily`(requer que você especifique `spec.minute` e `spec.hour`)

- `Weekly`(requer que você especifique `spec.minute, spec.hour` , e `spec.dayOfWeek`)
- `Monthly`(requer que você especifique `spec.minute, spec.hour` , e `spec.dayOfMonth`)
- `Custom`

◦ **spec.dayOfMonth**: (*Opcional*) O dia do mês (1 - 31) em que o agendamento deve ser executado. Este campo é obrigatório se a granularidade estiver definida como `Monthly` . O valor deve ser fornecido como uma string.

◦ **spec.dayOfWeek**: (*Opcional*) O dia da semana (0 - 7) em que a programação deve ser executada. Valores de 0 ou 7 indicam domingo. Este campo é obrigatório se a granularidade estiver definida como `Weekly` . O valor deve ser fornecido como uma string.

◦ **spec.hour**: (*Opcional*) A hora do dia (0 - 23) em que a programação deve ser executada. Este campo é obrigatório se a granularidade estiver definida como `Daily` , `Weekly` , ou `Monthly` . O valor deve ser fornecido como uma string.

◦ **spec.minute**: (*Opcional*) O minuto da hora (0 - 59) em que a programação deve ser executada. Este campo é obrigatório se a granularidade estiver definida como `Hourly` , `Daily` , `Weekly` , ou `Monthly` . O valor deve ser fornecido como uma string.

◦ **spec.runImmediately**: (*Opcional*) Defina como `true` para acionar uma execução única e imediata da linha de base (backup e/ou snapshot de acordo com as configurações de retenção) quando o agendamento for criado. O padrão é `false`. Isso não modifica a recorrência subsequente.

Exemplo de YAML para agendamento de backup e snapshot:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: Schedule
metadata:
  namespace: my-app-namespace
  name: my-cr-name
spec:
  dataMover: Kopia
  applicationRef: my-application
  appVaultRef: appvault-name
  backupRetention: "15"
  snapshotRetention: "15"
  granularity: Daily
  hour: "0"
  minute: "0"
```

Exemplo de YAML para programação somente de snapshot:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: Schedule
metadata:
```

```
  namespace: my-app-namespace
  name: my-snapshot-schedule
spec:
  applicationRef: my-application
  appVaultRef: appvault-name
  backupRetention: "0"
  snapshotRetention: "15"
  granularity: Daily
  hour: "2"
  minute: "0"
```

Exemplo de YAML para agendamento com execução imediata:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: Schedule
metadata:
  namespace: my-app-namespace
  name: my-daily-schedule-run-immediately
spec:
  applicationRef: my-application
  appVaultRef: appvault-name
  backupRetention: "7"
  snapshotRetention: "7"
  granularity: Daily
  hour: "3"
  minute: "0"
  runImmediately: true
```

3. Depois de preencher o `trident-protect-schedule-cr.yaml` ficheiro com os valores corretos, aplique o CR:

```
kubectl apply -f trident-protect-schedule-cr.yaml
```

Crie uma agenda usando a CLI

Passos

1. Crie o cronograma de proteção, substituindo valores entre parênteses por informações do seu ambiente. Por exemplo:

OBSERVAÇÃO	Você pode usar `tridentctl-protect create schedule --help` para exibir informações detalhadas de ajuda para este comando.

```
tridentctl-protect create schedule <my_schedule_name> \
```

```
  --appvault <my_appvault_name> \
  --app <name_of_app_to_snapshot> \
  --backup-retention <how_many_backups_to_retain> \
  --backup-reclaim-policy <Retain|Delete (default Retain)> \
  --data-mover <Kopia_or_Restic> \
  --day-of-month <day_of_month_to_run_schedule> \
  --day-of-week <day_of_week_to_run_schedule> \
  --granularity <frequency_to_run> \
  --hour <hour_of_day_to_run> \
  --minute <minute_of_hour_to_run> \
  --recurrence-rule <recurrence> \
  --snapshot-retention <how_many_snapshots_to_retain> \
  --snapshot-reclaim-policy <Retain|Delete (default Delete)> \
  --full-backup-rule <string> \
  --run-immediately <true|false> \
  -n <application_namespace>
```

As seguintes opções oferecem controle adicional sobre sua programação:

- **Agendamento completo de backups**: Use o `--full-backup-rule` Sinalizador para agendar backups completos não incrementais. Esta flag só funciona com `--granularity Daily`. Valores possíveis:
  - `Always`Faça um backup completo todos os dias.
  - Dias da semana específicos: Especifique um ou mais dias separados por vírgulas (por exemplo, `"Monday,Thursday"`). Valores válidos: segunda-feira, terça-feira, quarta-feira, quinta-feira, sexta-feira, sábado, domingo.

OBSERVAÇÃO	O `--full-backup-rule` A opção de sinalizar não funciona com granularidade horária, semanal ou mensal.

- **Proteção de linha de base imediata**: Use `--run-immediately true` para criar um backup ou snapshot inicial imediatamente quando o agendamento for criado, em vez de esperar pela primeira execução agendada. O padrão é `false`.
- **Agendamentos somente para instantâneos**: Definir `--backup-retention 0` e especifique um valor maior que zero para `--snapshot-retention`.

Anotações de agendamento suportadas

A tabela a seguir descreve as anotações que você pode usar ao criar uma solicitação de alteração (CR) de agendamento:

Anotação	Tipo	Descrição	Valor padrão
protect.trident.netapp.io/regra-de-backup-completo	cadeia de carateres	Especifica a regra para agendamento de backups completos. Você pode configurá-lo para `Always` Para backup completo constante ou personalize de acordo com suas necessidades. Por exemplo, se você escolher a granularidade diária, poderá especificar os dias da semana em que o backup completo deve ocorrer (por exemplo, `"Monday,Thursday"`). Os valores válidos para dias úteis são: segunda-feira, terça-feira, quarta-feira, quinta-feira, sexta-feira, sábado e domingo. Observe que esta anotação só pode ser usada com cronogramas que possuem `granularity` definido para `Daily`.	Não configurado (todos os backups são incrementais)
protect.trident.netapp.io/snapshot-completion-timeout	cadeia de carateres	Tempo máximo permitido para a conclusão da operação de captura instantânea.	"60m"
protect.trident.netapp.io/volume-snapshots-ready-to-use-timeout	cadeia de carateres	Tempo máximo permitido para que os snapshots de volume atinjam o estado pronto para uso.	"30m"
protect.trident.netapp.io/volume-snapshots-created-timeout	cadeia de carateres	Tempo máximo permitido para a criação de snapshots de volume.	"5m"
protect.trident.netapp.io/pvc-bind-timeout-sec	cadeia de carateres	Tempo máximo (em segundos) de espera para que quaisquer PersistentVolumeClaims (PVCs) recém-criados cheguem ao `Bound` fase anterior à falha das operações.	"1200" (20 minutos)

Eliminar um instantâneo

Exclua os snapshots programados ou sob demanda que você não precisa mais.

Passos

1. Remover o instantâneo CR associado ao instantâneo:

```
kubectl delete snapshot <snapshot_name> -n my-app-namespace
```

Eliminar uma cópia de segurança

Exclua os backups programados ou sob demanda que você não precisa mais.

OBSERVAÇÃO	Certifique-se de que a política de recuperação esteja definida como `Delete` para remover todos os dados de backup do armazenamento de objetos. A configuração padrão da política é `Retain` para evitar perda acidental de dados. Se a política não for alterada para `Delete` , os dados de backup permanecerão no armazenamento de objetos e exigirão exclusão manual.

Passos

1. Remova o CR de backup associado ao backup:

```
kubectl delete backup <backup_name> -n my-app-namespace
```

Verifique o status de uma operação de backup

Você pode usar a linha de comando para verificar o status de uma operação de backup em andamento, concluída ou falhou.

Passos

1. Use o seguinte comando para recuperar o status da operação de backup, substituindo valores em brackes por informações do seu ambiente:

```
kubectl get backup -n <namespace_name> <my_backup_cr_name> -o jsonpath
='{.status}'
```

Habilite o backup e a restauração de operações do Azure-NetApp-Files (ANF)

Se você instalou o Trident Protect, pode habilitar a funcionalidade de backup e restauração com uso eficiente de espaço para back-ends de armazenamento que utilizam a classe de armazenamento azure-netapp-files e foram criados antes do Trident 24.06. Essa funcionalidade opera com volumes NFSv4 e não consome espaço adicional do pool de capacidade.

Antes de começar

Certifique-se de que:

- Você instalou o Trident Protect.
- Você definiu um aplicativo no Trident Protect. Este aplicativo terá funcionalidade de proteção limitada até que você conclua este procedimento.
- Você `azure-netapp-files` selecionou como a classe de armazenamento padrão para o back-end de armazenamento.

▼ Expanda para obter as etapas de configuração

1. No Trident, se o volume do ANF tiver sido criado antes da atualização para o Trident 24,10:

   a. Ative o diretório instantâneo para cada PV que é baseado em azure-NetApp-Files e associado ao aplicativo:

   ```
   tridentctl update volume <pv name> --snapshot-dir=true -n trident
   ```

   b. Confirme se o diretório instantâneo foi ativado para cada PV associado:

   ```
   tridentctl get volume <pv name> -n trident -o yaml | grep
   snapshotDir
   ```

   Resposta:

```
snapshotDirectory: "true"
```

+
Quando o diretório de instantâneos não está habilitado, o Trident Protect escolhe a funcionalidade de backup regular, que consome temporariamente espaço no pool de capacidade durante o processo de backup. Nesse caso, certifique-se de que haja espaço suficiente disponível no pool de capacidade para criar um volume temporário do tamanho do volume que está sendo copiado.

Resultado

O aplicativo está pronto para backup e restauração usando o Trident Protect. Cada PVC também pode ser usado por outros aplicativos para backups e restaurações.

Use snapshots de grupo de volume com Trident Protect

Use snapshots de grupo de volume para capturar as declarações de volume persistente (PVCs) relacionadas no mesmo ponto no tempo. Isso é útil para aplicativos que gravam em mais de um volume, incluindo máquinas virtuais (VMs) com vários discos.

Sobre esta tarefa

Você define grupos de volume no `spec.volumeGroups` campo do `Application` recurso personalizado (CR). Quando você cria um snapshot da aplicação, o Trident Protect usa um grupo de consistência do ONTAP para criar snapshots de todos os PVCs em um grupo de volume no mesmo momento. Ele cria um grupo de consistência para cada namespace no grupo de volume. Cada volume no grupo também recebe seu próprio snapshot, para que você possa restaurar volumes individualmente.

O Trident Protect cria instantâneos individuais dos PVCs que não estão em um grupo de volume. Esses instantâneos são criados um após o outro, portanto, não há garantia de que reflitam o mesmo ponto no tempo.

Os snapshots de grupo de volume se aplicam a snapshots sob demanda, snapshots agendados e ao snapshot que o Trident Protect cria como parte de um backup. Se você fizer backup de um snapshot existente, o Trident Protect usará esse snapshot e não criará um novo snapshot de grupo. A replicação também não cria outro snapshot de grupo. Ela usa os snapshots por volume que o snapshot de grupo já criou.

O Trident Protect captura volumes no mesmo ponto no tempo somente se eles atenderem a todas as seguintes condições:

- Eles pertencem ao mesmo grupo de volume.
- Eles estão no mesmo namespace.
- Eles estão na mesma máquina virtual de armazenamento (SVM).
- O grupo de volume não possui mais de 80 volumes nesse namespace.

Se o Trident Protect não conseguir criar um snapshot de grupo, por exemplo, porque um requisito não foi atendido, ele adiciona um aviso a `status.warnings` e cria snapshots individuais dos PVCs desse grupo. Esses snapshots são criados um após o outro, portanto podem não refletir o mesmo ponto no tempo. O snapshot da aplicação ainda é concluído.

A tabela a seguir descreve o que acontece em situações comuns.

Situação	Resultado
O grupo de volume seleciona PVCs em mais de um espaço de nomes	Um snapshot de grupo pertence a um namespace, portanto o Trident Protect cria um snapshot de grupo separado em cada namespace. Esses snapshots de grupo não são obtidos no mesmo momento. Um pode ser bem-sucedido enquanto outro ainda está em execução, falha ou recorre a snapshots individuais. Por exemplo, PVCs rotulados `app: payments-db` em `payments-a` e `payments-b` tornam-se dois snapshots de grupo separados. Não trate snapshots de grupo de namespaces diferentes como um único ponto de restauração.
Os PVCs em um namespace estão em mais de uma SVM	O Trident Protect não divide volumes por SVM ou por backend do Trident. Um grupo de consistência do ONTAP não pode incluir volumes de mais de uma SVM, portanto, o snapshot do grupo falha para todo o namespace, incluindo os volumes que compartilham uma SVM. O Trident Protect adiciona um aviso, remove o snapshot do grupo e cria um snapshot individual de cada volume. Para capturar volumes juntos, coloque-os na mesma SVM.
Um namespace possui mais de 80 PVCs correspondentes	O Trident Protect ignora apenas esse namespace e cria snapshots individuais de seus PVCs. Outros namespaces no mesmo grupo de volume ainda recebem um snapshot de grupo.
Um PVC corresponde a mais de uma `volumeGroups` entrada	O Trident Protect adiciona o PVC a ambos os grupos de volume, e um dos snapshots do grupo pode falhar. O snapshot do grupo que falha reverte para snapshots individuais. Certifique-se de que cada PVC corresponda a apenas uma entrada.
Um grupo de volume inclui PVCs que usam um driver ou protocolo que não oferece suporte a snapshots de grupo de volume	A captura instantânea do grupo falha, e o Trident Protect cria capturas instantâneas individuais desses PVCs.

Antes de começar

- Seus PVCs devem usar um driver e um protocolo de armazenamento que sejam compatíveis com snapshots de grupo de volume. Para os drivers e protocolos compatíveis, consulte "Trabalhe com snapshots de grupo de volume do Trident".
- Seu cluster deve executar o Kubernetes 1.36 ou posterior.
- O controlador de snapshots do Kubernetes deve ser da versão 8.5.0 ou posterior, com o `CSIVolumeGroupSnapshot` feature gate ativado.
- Os `v1` CRDs para `VolumeGroupSnapshot`, `VolumeGroupSnapshotClass` e `VolumeGroupSnapshotContent` devem ser instalados.
- Você deve usar o Trident 26.10 ou posterior com um backend online. O `CSIVolumeGroupSnapshot` feature gate também deve estar habilitado no contêiner do Trident `csi-snapshotter`.
- Uma `VolumeGroupSnapshotClass`que usa o driver `csi.trident.netapp.io`deve ser definida. Se apenas uma classe for definida, o Trident Protect a usará. Se várias classes forem definidas, anote a classe que o Trident Protect deve usar com `snapshot.storage.kubernetes.io/is-default-class: "true"`. Caso contrário, o Trident Protect usará a primeira classe em ordem alfabética. Você não pode escolher a classe no `Application`ou no `Snapshot`CR. Para obter mais informações, consulte"Classe de instantâneo de grupo de volume".
- A conta do ONTAP no backend do Trident deve ter permissão para criar, obter e excluir grupos de

consistência, bem como para criar snapshots deles. Consulte "Gerenciar grupos de consistência de aplicativos" e "Gerenciar snapshots de grupo de consistência de aplicativos". A função padrão de administrador da SVM (`vsadmin`) não inclui essas permissões.

Defina um grupo de volume

O Trident Protect cria snapshots de grupo apenas para volumes que pertencem a um grupo de volume. Adicione `spec.volumeGroups` ao `Application` CR para definir os grupos. Cada entrada é um grupo separado. Selecione os volumes por rótulo, por VM ou por ambos.

Sobre esta tarefa

- Você só pode definir grupos de volume no `Application`CR. A `tridentctl-protect`CLI não tem uma opção de grupo de volume.
- O Trident Protect inclui apenas os PVCs vinculados em um grupo de volume e cria snapshots individuais dos PVCs que não estão vinculados. Se uma entrada não selecionar nenhum PVC vinculado, o Trident Protect não cria um snapshot de grupo para essa entrada.
- Certifique-se de que cada PVC corresponda a apenas uma entrada.

Antes de começar

Defina a aplicação que deseja proteger. Consulte "Defina uma aplicação para gestão com o Trident Protect.".

Passos

1. Se você selecionar volumes por rótulo, rotule os PVCs que deseja agrupar. Você pode usar qualquer chave e valor de rótulo. Os exemplos nesta página usam `volume-group=my-volume-group`.

```
kubectl label pvc <pvc_name_1> <pvc_name_2> volume-group=my-volume-group
-n <application_namespace>
```

2. Adicione `spec.volumeGroups` ao seu arquivo CR existente `Application`. Cada entrada deve incluir `labelSelector<volume_group>, vmSelector<volume_group>` ou ambos. Se uma entrada incluir ambos os campos, o grupo de volume incluirá todos os volumes que qualquer um dos campos selecionar.

   - **spec.volumeGroups[].labelSelector**: um seletor de rótulo do Kubernetes (`matchLabels` ou `matchExpressions`) que seleciona PVCs na aplicação. Um seletor vazio não é válido.
   - **spec.volumeGroups[].vmSelector**: uma lista de KubeVirt VMs, cada uma com um `namespace` e `name`. O Trident Protect inclui os PVCs que cada VM monta.

     O exemplo a seguir define dois grupos de volume. O primeiro grupo inclui os PVCs rotulados como `volume-group: my-volume-group`. O segundo grupo inclui os PVCs que `my-vm-1` e `my-vm-2` montam.

```
apiVersion: protect.trident.netapp.io/v1
kind: Application
metadata:
  name: my-app-name
  namespace: my-namespace
spec:
  includedNamespaces:
```

```
    - namespace: my-namespace
  volumeGroups:
    - labelSelector:
        matchLabels:
          volume-group: my-volume-group
    - vmSelector:
        - namespace: my-namespace
          name: my-vm-1
        - namespace: my-namespace
          name: my-vm-2
```

+

OBSERVAÇÃO	Mantenha os outros campos que seu `Application` CR já contém, como `spec.resourceFilter`. Se você omitir um campo do arquivo que aplica, `kubectl apply` o exclui do CR.

3. Aplique a CR atualizada:

```
kubectl apply -f <application_cr_file>.yaml
```

Resultado

O `Application` CR inclui os grupos de volume. O próximo snapshot da aplicação os utiliza.

Criar um instantâneo

Crie um snapshot sob demanda, um backup sob demanda ou um agendamento de proteção de dados para o aplicativo. Você não precisa adicionar nenhum campo. O Trident Protect usa os grupos de volume que você definiu no `Application` CR. Para obter instruções, consulte "Proteja aplicativos usando o Trident Protect.".

OBSERVAÇÃO	Por padrão, o Trident Protect cria o snapshot do grupo antes de congelar os sistemas de arquivos da VM. Os volumes em um grupo de volume são capturados sem congelamento, portanto os dados que uma VM ainda não gravou em disco não são incluídos. Os discos que não estão no grupo de volume são capturados depois, quando o Trident Protect cria snapshots individuais após congelar a VM. Para capturar todos os discos de uma VM no mesmo momento, inclua todos os discos no grupo de volume. Para obter mais informações sobre o congelamento de sistemas de arquivos da VM, consulte "Proteção de dados com máquinas virtuais do KubeVirt".

Analise o resultado

Após a conclusão do snapshot, verifique o status para confirmar se o Trident Protect criou os snapshots de grupo.

OBSERVAÇÃO	Um snapshot de aplicação concluído não significa que seus snapshots de grupo foram bem-sucedidos. A `VolumeGroupSnapshotsCreated` condição persiste `true` mesmo se todos os snapshots de grupo forem ignorados ou falharem, e o

> Trident Protect não criar um evento ou métrica no Kubernetes. Para detectar isso, monitore `status.warnings` e os campos `skipped` e `failed`.

Passos

1. Obtenha o nome do snapshot. Para listar os snapshots da aplicação, execute o seguinte comando:

```
kubectl get snapshots -n <application_namespace>
```

2. Verifique o status do instantâneo:

```
kubectl get snapshots <snapshot_name> -n <application_namespace> -o yaml
```

Resultado

O `status.volumeGroupSnapshots` campo possui uma entrada para cada snapshot de grupo. O Trident Protect cria um snapshot de grupo separado para cada namespace em um grupo de volume. O exemplo a seguir está abreviado:

```
status:
  volumeGroupSnapshots:
    - name: vgs-<snapshot_uid_prefix>-<group_index>-<namespace>
      namespace: <namespace>
      ready: true
      memberPVCs:
        - pvcName: <pvc_name_1>
          pvcNamespace: <namespace>
        - pvcName: <pvc_name_2>
          pvcNamespace: <namespace>
```

O nome do snapshot do grupo inclui os primeiros oito caracteres do `Snapshot` UID, o índice da `spec.volumeGroups` entrada (começando em `0`) e o namespace, truncado para 12 caracteres.

- `ready: true` significa que o snapshot do grupo foi bem-sucedido. Os snapshots que compõem o grupo também estão listados em `status.volumeSnapshots`. Estes são os snapshots dos membros do grupo, não snapshots adicionais dos mesmos volumes.
- `skipped: true`significa que o Trident Protect não tentou criar o snapshot do grupo, porque o grupo de volume tem mais de 80 PVCs nesse namespace. O Trident Protect cria snapshots individuais dos volumes do grupo em vez disso e adiciona o `skipReason`texto a `status.warnings`.
- `failed: true` significa que o Trident Protect tentou criar o snapshot do grupo, mas falhou. O Trident Protect cria snapshots individuais dos volumes do grupo em vez disso e adiciona o `error`texto a `status.warnings`.

Etiquetas temporárias de PVC e limpeza

O Trident Protect adiciona um rótulo a cada PVC em um snapshot de grupo, por exemplo `protect.trident.netapp.io/vgs-group-673c0189-0=true`. Seus próprios rótulos não são

alterados. O rótulo permanece no PVC até que você exclua o snapshot da aplicação. Quando você exclui o snapshot da aplicação, o Trident Protect também exclui seus snapshots de grupo de volume e snapshots de membro, além de remover os rótulos dos PVCs. Isso acontece independentemente de o `reclaimPolicy` do snapshot ser `Delete` ou `Retain`.

O que se segue?

Restaure o aplicativo a partir do snapshot da mesma forma que para qualquer outro snapshot. Os snapshots de grupos de volume não têm restauração separada. O Trident Protect restaura a partir dos snapshots dos membros, para que você possa restaurar PVCs ou VMs individuais sem restaurar todos os PVCs do grupo. Consulte "Restauração de um snapshot para um namespace diferente" ou "Restauração de um snapshot para o namespace original".

Informações relacionadas

- "Defina uma aplicação para gestão com o Trident Protect."
- "Proteja aplicativos usando o Trident Protect."
- "Proteja máquinas virtuais KubeVirt com Trident Protect"

Restaurar aplicativos

Restaure aplicativos usando o Trident Protect.

Você pode usar o Trident Protect para restaurar seu aplicativo a partir de um snapshot ou backup. A restauração a partir de um snapshot existente será mais rápida ao restaurar o aplicativo no mesmo cluster.

OBSERVAÇÃO

- Quando você restaura um aplicativo, todos os ganchos de execução configurados para o aplicativo são restaurados com o aplicativo. Se um gancho de execução pós-restauração estiver presente, ele será executado automaticamente como parte da operação de restauração.
- A restauração de um backup para um namespace diferente ou para o namespace original é suportada para volumes qtree. No entanto, a restauração de um snapshot para um namespace diferente ou para o namespace original não é suportada para volumes qtree.
- Você pode usar configurações avançadas para personalizar as operações de restauração. Para saber mais, consulte "Utilize as configurações avançadas de restauração do Trident Protect.".
- Para restaurar máquinas virtuais (VMs) específicas do KubeVirt a partir de um arquivo sem restaurar todos os recursos do namespace, use o campo `includedVirtualMachines` em um CR de restauração. Para CRs compatíveis, referência de campos e exemplos de YAML, consulte "Proteja máquinas virtuais KubeVirt com Trident Protect".

Restaure de um backup para um namespace diferente

Ao restaurar um backup para um namespace diferente usando um CR de BackupRestore, o Trident Protect restaura o aplicativo em um novo namespace e cria um CR de aplicativo para o aplicativo restaurado. Para proteger a aplicação restaurada, crie backups ou snapshots sob demanda, ou estabeleça um cronograma de proteção.

OBSERVAÇÃO	• Restaurar um backup para um namespace diferente com recursos existentes não alterará nenhum recurso que compartilhe nomes com aqueles no backup. Para restaurar todos os recursos no backup, exclua e recrie o namespace de destino ou restaure o backup para um novo namespace. • Ao usar uma solicitação de restauração (CR) para restaurar em um novo namespace, você deve criar manualmente o namespace de destino antes de aplicar a CR. O Trident Protect cria namespaces automaticamente apenas quando se utiliza a CLI (linha de comando).

Antes de começar

Certifique-se de que a expiração do token de sessão da AWS seja suficiente para quaisquer operações de restauração S3 de longa duração. Se o token expirar durante a operação de restauração, a operação pode falhar.

- Consulte a "Documentação da API da AWS" para obter mais informações sobre como verificar a expiração do token de sessão atual.
- Consulte o "Documentação do AWS IAM" para obter mais informações sobre credenciais com recursos da AWS.

OBSERVAÇÃO	Ao restaurar backups usando o Kopia como o movimentador de dados, você pode, opcionalmente, especificar anotações no CR ou usar a CLI para controlar o comportamento do armazenamento temporário usado pelo Kopia. Consulte o "Documentação da Kopia" Para obter mais informações sobre as opções que você pode configurar. Use o `tridentctl-protect create --help` Para obter mais informações sobre como especificar anotações com a CLI do Trident Protect, consulte o comando.

Use um CR

Passos

1. Crie o arquivo de recurso personalizado (CR) e nomeie-o `trident-protect-backup-restore-cr.yaml`.
2. No arquivo criado, configure os seguintes atributos:
   - **metadata.name**: (*required*) o nome deste recurso personalizado; escolha um nome único e sensível para o seu ambiente.
   - **Spec.appArchivePath**: O caminho dentro do AppVault onde o conteúdo do backup é armazenado. Você pode usar o seguinte comando para encontrar este caminho:

     ```
     kubectl get backups <BACKUP_NAME> -n my-app-namespace -o jsonpath='{.status.appArchivePath}'
     ```

   - **Spec.appVaultRef**: (*required*) o nome do AppVault onde o conteúdo de backup é armazenado.
   - **spec.destinationApplicationName**: (*Opcional*) O nome para o aplicativo restaurado. Se fornecido, o aplicativo restaurado usa este nome. Se não for fornecido, o aplicativo restaurado usa o nome do aplicativo de origem.
   - **spec.namespaceMapping**: o mapeamento do namespace de origem da operação de restauração para o namespace de destino. Substitua `my-source-namespace` e `my-`

`destination-namespace` por informações do seu ambiente.

```
---
apiVersion: protect.trident.netapp.io/v1
kind: BackupRestore
metadata:
  name: my-cr-name
  namespace: my-destination-namespace
spec:
  appArchivePath: my-backup-path
  appVaultRef: appvault-name
  destinationApplicationName: my-new-app-name
  namespaceMapping: [{"source": "my-source-namespace",
"destination": "my-destination-namespace"}]
```

3. (*Opcional*) se você precisar selecionar apenas determinados recursos do aplicativo para restaurar, adicione filtragem que inclua ou exclua recursos marcados com rótulos específicos:

OBSERVAÇÃO	O Trident Protect seleciona alguns recursos automaticamente devido à sua relação com recursos que você seleciona. Por exemplo, se você selecionar um recurso de reivindicação de volume persistente e ele tiver um pod associado, o Trident Protect também restaurará o pod associado.

   - **ResourceFilter.resourceSelectionCriteria**: (Necessário para filtragem) Use `Include` ou `Exclude` inclua ou exclua um recurso definido em resourceMatchers. Adicione os seguintes parâmetros resourceMatchers para definir os recursos a serem incluídos ou excluídos:
     - **ResourceFilter.resourceMatchers**: Uma matriz de resourceMatcher objetos. Se você definir vários elementos nesse array, eles corresponderão como uma OPERAÇÃO OU, e os campos dentro de cada elemento (grupo, tipo, versão) corresponderão como uma OPERAÇÃO E.
       - **ResourceMatchers[].group**: (*Optional*) Grupo do recurso a ser filtrado.
       - **ResourceMatchers[].kind**: (*Opcional*) tipo do recurso a ser filtrado.
       - **ResourceMatchers[].version**: (*Optional*) versão do recurso a ser filtrado.
       - **ResourceMatchers[].names**: (*Optional*) nomes no campo Kubernetes metadata.name do recurso a ser filtrado.
       - **ResourceMatchers[].namespaces**: (*Optional*) namespaces no campo Kubernetes metadata.name do recurso a ser filtrado.
       - **ResourceMatchers[].labelSelectors**: (*Optional*) string de seleção de etiquetas no campo Kubernetes metadata.name do recurso, conforme definido no "Documentação do Kubernetes". Por exemplo `"trident.netapp.io/os=linux"`:.

         Por exemplo:

```
spec:
  resourceFilter:
    resourceSelectionCriteria: "Include"
```

```
    resourceMatchers:
      - group: my-resource-group-1
        kind: my-resource-kind-1
        version: my-resource-version-1
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
      - group: my-resource-group-2
        kind: my-resource-kind-2
        version: my-resource-version-2
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
```

4. Depois de preencher o `trident-protect-backup-restore-cr.yaml` ficheiro com os valores corretos, aplique o CR:

```
kubectl apply -f trident-protect-backup-restore-cr.yaml
```

Use a CLI

Passos

1. Restaure o backup para um namespace diferente, substituindo valores entre parênteses por informações do seu ambiente. O `namespace-mapping` argumento usa namespaces separados por dois pontos para mapear namespaces de origem para os namespaces de destino corretos no formato `source1:dest1,source2:dest2`. Por exemplo:

```
tridentctl-protect create backuprestore <my_restore_name> \
--backup <backup_namespace>/<backup_to_restore> \
--namespace-mapping <source_to_destination_namespace_mapping> \
--destination-app-name<custom_app_name>\
-n <application_namespace>
```

Restaure de um backup para o namespace original

Você pode restaurar um backup para o namespace original a qualquer momento. Ao realizar uma restauração no local, Trident Protect gerencia automaticamente os agendamentos de proteção e as operações em andamento para evitar pontos de recuperação inválidos:

- Todos os agendamentos de proteção ativados para o aplicativo são desativados antes do início da restauração. Isso impede que backups ou snapshots agendados sejam executados enquanto os recursos do aplicativo estão sendo restaurados.
- Após a restauração ser concluída com sucesso, somente os agendamentos que estavam ativados antes da restauração são reativados. Os agendamentos que já estavam desativados permanecem desativados.
- Quaisquer operações de backup ou snapshot em andamento são canceladas antes do início da restauração. Se uma operação não for cancelada em 5 minutos, a restauração prossegue e registra um

aviso no status do CR de restauração.

Antes de começar

Certifique-se de que a expiração do token de sessão da AWS seja suficiente para quaisquer operações de restauração S3 de longa duração. Se o token expirar durante a operação de restauração, a operação pode falhar.

- Consulte a "Documentação da API da AWS" para obter mais informações sobre como verificar a expiração do token de sessão atual.
- Consulte o "Documentação do AWS IAM" para obter mais informações sobre credenciais com recursos da AWS.

OBSERVAÇÃO	Ao restaurar backups usando o Kopia como o movimentador de dados, você pode, opcionalmente, especificar anotações no CR ou usar a CLI para controlar o comportamento do armazenamento temporário usado pelo Kopia. Consulte o "Documentação da Kopia" Para obter mais informações sobre as opções que você pode configurar. Use o `tridentctl-protect create --help` Para obter mais informações sobre como especificar anotações com a CLI do Trident Protect, consulte o comando.

Use um CR

Passos

1. Crie o arquivo de recurso personalizado (CR) e nomeie-o `trident-protect-backup-ipr-cr.yaml`.
2. No arquivo criado, configure os seguintes atributos:
   - **metadata.name**: (*required*) o nome deste recurso personalizado; escolha um nome único e sensível para o seu ambiente.
   - **Spec.appArchivePath**: O caminho dentro do AppVault onde o conteúdo do backup é armazenado. Você pode usar o seguinte comando para encontrar este caminho:

     ```
     kubectl get backups <BACKUP_NAME> -n my-app-namespace -o jsonpath='{.status.appArchivePath}'
     ```

   - **Spec.appVaultRef**: (*required*) o nome do AppVault onde o conteúdo de backup é armazenado.

     Por exemplo:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: BackupInplaceRestore
metadata:
  name: my-cr-name
  namespace: my-app-namespace
spec:
  appArchivePath: my-backup-path
```

```
  appVaultRef: appvault-name
```

3. (*Opcional*) se você precisar selecionar apenas determinados recursos do aplicativo para restaurar, adicione filtragem que inclua ou exclua recursos marcados com rótulos específicos:

OBSERVAÇÃO	O Trident Protect seleciona alguns recursos automaticamente devido à sua relação com recursos que você seleciona. Por exemplo, se você selecionar um recurso de reivindicação de volume persistente e ele tiver um pod associado, o Trident Protect também restaurará o pod associado.

   - **ResourceFilter.resourceSelectionCriteria**: (Necessário para filtragem) Use `Include` ou `Exclude` inclua ou exclua um recurso definido em resourceMatchers. Adicione os seguintes parâmetros resourceMatchers para definir os recursos a serem incluídos ou excluídos:
     - **ResourceFilter.resourceMatchers**: Uma matriz de resourceMatcher objetos. Se você definir vários elementos nesse array, eles corresponderão como uma OPERAÇÃO OU, e os campos dentro de cada elemento (grupo, tipo, versão) corresponderão como uma OPERAÇÃO E.
       - **ResourceMatchers[].group**: (*Optional*) Grupo do recurso a ser filtrado.
       - **ResourceMatchers[].kind**: (*Opcional*) tipo do recurso a ser filtrado.
       - **ResourceMatchers[].version**: (*Optional*) versão do recurso a ser filtrado.
       - **ResourceMatchers[].names**: (*Optional*) nomes no campo Kubernetes metadata.name do recurso a ser filtrado.
       - **ResourceMatchers[].namespaces**: (*Optional*) namespaces no campo Kubernetes metadata.name do recurso a ser filtrado.
       - **ResourceMatchers[].labelSelectors**: (*Optional*) string de seleção de etiquetas no campo Kubernetes metadata.name do recurso, conforme definido no "Documentação do Kubernetes". Por exemplo `"trident.netapp.io/os=linux"`: .

         Por exemplo:

```
spec:
  resourceFilter:
    resourceSelectionCriteria: "Include"
    resourceMatchers:
      - group: my-resource-group-1
        kind: my-resource-kind-1
        version: my-resource-version-1
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
      - group: my-resource-group-2
        kind: my-resource-kind-2
        version: my-resource-version-2
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
```

4. Depois de preencher o `trident-protect-backup-ipr-cr.yaml` ficheiro com os valores corretos, aplique o CR:

```
kubectl apply -f trident-protect-backup-ipr-cr.yaml
```

Use a CLI

Passos

1. Restaure o backup para o namespace original, substituindo valores entre parênteses por informações do seu ambiente. O `backup` argumento usa um namespace e um nome de backup no formato `<namespace>/<name>`. Por exemplo:

```
tridentctl-protect create backupinplacerestore <my_restore_name> \
--backup <namespace/backup_to_restore> \
-n <application_namespace>
```

Restaure de um backup para um cluster diferente

Você pode restaurar um backup para um cluster diferente se houver um problema com o cluster original.

OBSERVAÇÃO

- Ao restaurar backups usando o Kopia como o movimentador de dados, você pode, opcionalmente, especificar anotações no CR ou usar a CLI para controlar o comportamento do armazenamento temporário usado pelo Kopia. Consulte o "Documentação da Kopia" Para obter mais informações sobre as opções que você pode configurar. Use o `tridentctl-protect create --help` Para obter mais informações sobre como especificar anotações com a CLI do Trident Protect, consulte o comando.
- Ao usar uma solicitação de restauração (CR) para restaurar em um novo namespace, você deve criar manualmente o namespace de destino antes de aplicar a CR. O Trident Protect cria namespaces automaticamente apenas quando se utiliza a CLI (linha de comando).

Antes de começar

Certifique-se de que os seguintes pré-requisitos são cumpridos:

- O cluster de destino tem o Trident Protect instalado.
- O cluster de destino tem acesso ao caminho do bucket do mesmo AppVault que o cluster de origem, onde o backup é armazenado.
- Ao executar o AppVault CR, certifique-se de que seu ambiente local possa se conectar ao bucket de armazenamento de objetos definido nele. `tridentctl-protect get appvaultcontent` comando. Caso as restrições de rede impeçam o acesso, execute a CLI do Trident Protect a partir de um pod no cluster de destino.
- Certifique-se de que a expiração do token de sessão da AWS seja suficiente para quaisquer operações de restauração de longa duração. Se o token expirar durante a operação de restauração, a operação pode falhar.
  - Consulte a "Documentação da API da AWS" para obter mais informações sobre como verificar a expiração do token de sessão atual.

- Consulte o "Documentação do AWS" para obter mais informações sobre credenciais com recursos da AWS.

Passos

1. Verifique se o AppVault CR existe no cluster de destino usando o plugin Trident Protect CLI:

```
tridentctl-protect get appvault --context <destination_cluster_name>
```

OBSERVAÇÃO	Se o AppVault CR não existir no cluster de destino, crie-o seguindo os passos em "Use objetos do Trident Protect AppVault para gerenciar buckets.".

2. Visualize o conteúdo do backup disponível do AppVault no cluster de destino e anote `appArchivePath` do backup que deseja restaurar:

```
tridentctl-protect get appvaultcontent <appvault_name> \
--show-resources backup \
--show-paths \
--context <destination_cluster_name>
```

Executar esse comando exibe os backups disponíveis no AppVault, incluindo os clusters de origem, nomes de aplicativos correspondentes, carimbos de data/hora e caminhos de arquivamento.

Exemplo de saída:

```
+-------------+-----------+--------+-----------------
+--------------------------+-------------+
|   CLUSTER   |    APP    |  TYPE  |      NAME       |         TIMESTAMP
|    PATH     |
+-------------+-----------+--------+-----------------
+--------------------------+-------------+
| production1 | wordpress | backup | wordpress-bkup-1| 2024-10-30
08:37:40 (UTC)| backuppath1 |
| production1 | wordpress | backup | wordpress-bkup-2| 2024-10-30
08:37:40 (UTC)| backuppath2 |
+-------------+-----------+--------+-----------------
+--------------------------+-------------+
```

3. Restaure o aplicativo para o cluster de destino usando o nome do AppVault e o caminho do arquivo:

OBSERVAÇÃO	Ao usar uma CR, certifique-se de que o namespace destinado à restauração do aplicativo exista no cluster de destino.

Use um CR

1. Crie o arquivo de recurso personalizado (CR) e nomeie-o `trident-protect-backup-restore-`

`cr.yaml`.

2. No arquivo criado, configure os seguintes atributos:

   - **metadata.name**: (*required*) o nome deste recurso personalizado; escolha um nome único e sensível para o seu ambiente.
   - **Spec.appVaultRef**: (*required*) o nome do AppVault onde o conteúdo de backup é armazenado.
   - **spec.appArchivePath**: (*Obrigatório*) O caminho dentro do AppVault onde o conteúdo do backup está armazenado. Use o comando da etapa 2 para visualizar o conteúdo do backup e encontrar `appArchivePath` o backup que deseja restaurar.
   - **spec.destinationApplicationName**: (*Opcional*) O nome para o aplicativo restaurado. Se fornecido, o aplicativo restaurado usa este nome. Se não for fornecido, o aplicativo restaurado usa o nome do aplicativo de origem.
   - **spec.namespaceMapping**: o mapeamento do namespace de origem da operação de restauração para o namespace de destino. Substitua `my-source-namespace` e `my-destination-namespace` por informações do seu ambiente.

     Por exemplo:

```
apiVersion: protect.trident.netapp.io/v1
kind: BackupRestore
metadata:
  name: my-cr-name
  namespace: my-destination-namespace
spec:
  appVaultRef: appvault-name
  appArchivePath: my-backup-path
  destinationApplicationName: my-new-app-name
  namespaceMapping: [{"source": "my-source-namespace", "
destination": "my-destination-namespace"}]
```

3. Depois de preencher o `trident-protect-backup-restore-cr.yaml` ficheiro com os valores corretos, aplique o CR:

```
kubectl apply -f trident-protect-backup-restore-cr.yaml
```

Use a CLI

1. Use o comando a seguir para restaurar o aplicativo, substituindo valores entre parênteses por informações do ambiente. O argumento namespace-mapping usa namespaces separados por dois pontos para mapear namespaces de origem para os namespaces de destino corretos no formato source1:dest1,source2:dest2. Por exemplo:

```
tridentctl-protect create backuprestore <restore_name> \
--namespace-mapping <source_to_destination_namespace_mapping> \
--appvault <appvault_name> \
--path <backup_path> \
```

```
--destination-app-name <custom_app_name> \
--context <destination_cluster_name> \
-n <application_namespace>
```

Restauração de um snapshot para um namespace diferente

Você pode restaurar dados de um snapshot usando um arquivo de recurso personalizado (CR) para um namespace diferente ou para o namespace de origem original. Ao restaurar um snapshot para um namespace diferente usando um CR de SnapshotRestore, o Trident Protect restaura o aplicativo em um novo namespace e cria um CR de aplicativo para o aplicativo restaurado. Para proteger a aplicação restaurada, crie backups ou snapshots sob demanda, ou estabeleça um cronograma de proteção.

OBSERVAÇÃO

- O SnapshotRestore oferece suporte ao `spec.storageClassMapping` atributo, mas somente quando as classes de armazenamento de origem e destino usam o mesmo backend de armazenamento. Se você tentar restaurar para um `StorageClass` que usa um backend de armazenamento diferente, a operação de restauração falhará.
- Ao usar uma solicitação de restauração (CR) para restaurar em um novo namespace, você deve criar manualmente o namespace de destino antes de aplicar a CR. O Trident Protect cria namespaces automaticamente apenas quando se utiliza a CLI (linha de comando).

Antes de começar

Certifique-se de que a expiração do token de sessão da AWS seja suficiente para quaisquer operações de restauração S3 de longa duração. Se o token expirar durante a operação de restauração, a operação pode falhar.

- Consulte a "Documentação da API da AWS" para obter mais informações sobre como verificar a expiração do token de sessão atual.
- Consulte o "Documentação do AWS IAM" para obter mais informações sobre credenciais com recursos da AWS.

Use um CR

Passos

1. Crie o arquivo de recurso personalizado (CR) e nomeie-o `trident-protect-snapshot-restore-cr.yaml`.
2. No arquivo criado, configure os seguintes atributos:
   - **metadata.name**: (*required*) o nome deste recurso personalizado; escolha um nome único e sensível para o seu ambiente.
   - **Spec.appVaultRef**: (*required*) o nome do AppVault onde o conteúdo do instantâneo é armazenado.
   - **Spec.appArchivePath**: O caminho dentro do AppVault onde o conteúdo do snapshot é armazenado. Você pode usar o seguinte comando para encontrar este caminho:

     ```
     kubectl get snapshots <SNAPHOT_NAME> -n my-app-namespace -o
     ```

```
jsonpath='{.status.appArchivePath}'
```

- **spec.destinationApplicationName**: (*Opcional*) O nome para o aplicativo restaurado. Se fornecido, o aplicativo restaurado usa este nome. Se não for fornecido, o aplicativo restaurado usa o nome do aplicativo de origem.
- **spec.namespaceMapping**: o mapeamento do namespace de origem da operação de restauração para o namespace de destino. Substitua `my-source-namespace` e `my-destination-namespace` por informações do seu ambiente.

```
---
apiVersion: protect.trident.netapp.io/v1
kind: SnapshotRestore
metadata:
  name: my-cr-name
  namespace: my-destination-namespace
spec:
  appVaultRef: appvault-name
  appArchivePath: my-snapshot-path
  namespaceMapping: [{"source": "my-source-namespace",
"destination": "my-destination-namespace"}]
```

3. (*Opcional*) se você precisar selecionar apenas determinados recursos do aplicativo para restaurar, adicione filtragem que inclua ou exclua recursos marcados com rótulos específicos:

OBSERVAÇÃO	O Trident Protect seleciona alguns recursos automaticamente devido à sua relação com recursos que você seleciona. Por exemplo, se você selecionar um recurso de reivindicação de volume persistente e ele tiver um pod associado, o Trident Protect também restaurará o pod associado.

- **ResourceFilter.resourceSelectionCriteria**: (Necessário para filtragem) Use `Include` ou `Exclude` inclua ou exclua um recurso definido em resourceMatchers. Adicione os seguintes parâmetros resourceMatchers para definir os recursos a serem incluídos ou excluídos:
  - **ResourceFilter.resourceMatchers**: Uma matriz de resourceMatcher objetos. Se você definir vários elementos nesse array, eles corresponderão como uma OPERAÇÃO OU, e os campos dentro de cada elemento (grupo, tipo, versão) corresponderão como uma OPERAÇÃO E.
    - **ResourceMatchers[].group**: (*Optional*) Grupo do recurso a ser filtrado.
    - **ResourceMatchers[].kind**: (*Opcional*) tipo do recurso a ser filtrado.
    - **ResourceMatchers[].version**: (*Optional*) versão do recurso a ser filtrado.
    - **ResourceMatchers[].names**: (*Optional*) nomes no campo Kubernetes metadata.name do recurso a ser filtrado.
    - **ResourceMatchers[].namespaces**: (*Optional*) namespaces no campo Kubernetes metadata.name do recurso a ser filtrado.
    - **ResourceMatchers[].labelSelectors**: (*Optional*) string de seleção de etiquetas no campo Kubernetes metadata.name do recurso, conforme definido no "Documentação do Kubernetes". Por exemplo `"trident.netapp.io/os=linux"`: .

Por exemplo:

```
spec:
  resourceFilter:
    resourceSelectionCriteria: "Include"
    resourceMatchers:
      - group: my-resource-group-1
        kind: my-resource-kind-1
        version: my-resource-version-1
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
      - group: my-resource-group-2
        kind: my-resource-kind-2
        version: my-resource-version-2
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
```

4. Depois de preencher o `trident-protect-snapshot-restore-cr.yaml` ficheiro com os valores corretos, aplique o CR:

```
kubectl apply -f trident-protect-snapshot-restore-cr.yaml
```

Use a CLI

Passos

1. Restaure o snapshot para um namespace diferente, substituindo valores entre parênteses por informações do seu ambiente.

   - O `snapshot` argumento usa um namespace e um nome instantâneo no formato `<namespace>/<name>`.
   - O `namespace-mapping` argumento usa namespaces separados por dois pontos para mapear namespaces de origem para os namespaces de destino corretos no formato `source1:dest1,source2:dest2`.

   Por exemplo:

```
tridentctl-protect create snapshotrestore <my_restore_name> \
--snapshot <namespace/snapshot_to_restore> \
--namespace-mapping <source_to_destination_namespace_mapping> \
--destination-app-name <custom_app_name> \
-n <application_namespace>
```

Restauração de um snapshot para o namespace original

Você pode restaurar um snapshot para o namespace original a qualquer momento. Ao realizar uma restauração no local, Trident Protect gerencia automaticamente os agendamentos de proteção e as operações em andamento para evitar pontos de recuperação inválidos:

- Todos os agendamentos de proteção ativados para o aplicativo são desativados antes do início da restauração. Isso impede que backups ou snapshots agendados sejam executados enquanto os recursos do aplicativo estão sendo restaurados.
- Após a restauração ser concluída com sucesso, somente os agendamentos que estavam ativados antes da restauração são reativados. Os agendamentos que já estavam desativados permanecem desativados.
- Quaisquer operações de backup ou snapshot em andamento são canceladas antes do início da restauração. Se uma operação não for cancelada em 5 minutos, a restauração prossegue e registra um aviso no status do CR de restauração.

Antes de começar

Certifique-se de que a expiração do token de sessão da AWS seja suficiente para quaisquer operações de restauração S3 de longa duração. Se o token expirar durante a operação de restauração, a operação pode falhar.

- Consulte a "Documentação da API da AWS" para obter mais informações sobre como verificar a expiração do token de sessão atual.
- Consulte o "Documentação do AWS IAM" para obter mais informações sobre credenciais com recursos da AWS.

Use um CR

Passos

1. Crie o arquivo de recurso personalizado (CR) e nomeie-o `trident-protect-snapshot-ipr-cr.yaml`.
2. No arquivo criado, configure os seguintes atributos:
   - **metadata.name**: (*required*) o nome deste recurso personalizado; escolha um nome único e sensível para o seu ambiente.
   - **Spec.appVaultRef**: (*required*) o nome do AppVault onde o conteúdo do instantâneo é armazenado.
   - **Spec.appArchivePath**: O caminho dentro do AppVault onde o conteúdo do snapshot é armazenado. Você pode usar o seguinte comando para encontrar este caminho:

     ```
     kubectl get snapshots <SNAPSHOT_NAME> -n my-app-namespace -o jsonpath='{.status.appArchivePath}'
     ```

```
---
apiVersion: protect.trident.netapp.io/v1
kind: SnapshotInplaceRestore
metadata:
  name: my-cr-name
  namespace: my-app-namespace
```

```
spec:
  appVaultRef: appvault-name
  appArchivePath: my-snapshot-path
```

+ . (Opcional) Se você precisar selecionar apenas determinados recursos do aplicativo para restaurar, adicione filtros que incluam ou excluam recursos marcados com rótulos específicos:

+ NOTA: Trident Protect seleciona alguns recursos automaticamente devido à sua relação com os recursos que você seleciona. Por exemplo, se você selecionar um recurso de reivindicação de volume persistente e ele tiver um pod associado, Trident Protect também restaurará o pod associado.

+ * **resourceFilter.resourceSelectionCriteria**: (Obrigatório para filtragem) Use `Include` ou `Exclude` para incluir ou excluir um recurso definido em resourceMatchers. Adicione os seguintes parâmetros resourceMatchers para definir os recursos a serem incluídos ou excluídos: **resourceFilter.resourceMatchers: Uma matriz de objetos resourceMatcher. Se você definir vários elementos nesta matriz, eles correspondem como uma operação OR, e os campos dentro de cada elemento (group, kind, version) correspondem como uma operação AND.** * **resourceMatchers[].group**: (Opcional) Grupo do recurso a ser filtrado. *** resourceMatchers[].kind: (Opcional) Tipo do recurso a ser filtrado.** * **resourceMatchers[].version**: (Opcional) Versão do recurso a ser filtrado. *** resourceMatchers[].names: (Opcional) Nomes no campo metadata.name do Kubernetes do recurso a ser filtrado.** * **resourceMatchers[].namespaces**: (Opcional) Namespaces no campo metadata.name do Kubernetes do recurso a ser filtrado. *** resourceMatchers[].labelSelectors**: (Opcional) String seletora de rótulo no campo metadata.name do Kubernetes do recurso, conforme definido em "Documentação do Kubernetes". Por exemplo: `"trident.netapp.io/os=linux"`.

+ Por exemplo:

+

```
spec:
  resourceFilter:
    resourceSelectionCriteria: "Include"
    resourceMatchers:
      - group: my-resource-group-1
        kind: my-resource-kind-1
        version: my-resource-version-1
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
      - group: my-resource-group-2
        kind: my-resource-kind-2
        version: my-resource-version-2
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
```

+ . Depois de preencher o `trident-protect-snapshot-ipr-cr.yaml` file com os valores corretos, aplique a CR:

+

```
kubectl apply -f trident-protect-snapshot-ipr-cr.yaml
```

Use a CLI

Passos

1. Restaure o snapshot para o namespace original, substituindo valores entre parênteses por informações do seu ambiente. Por exemplo:

```
tridentctl-protect create snapshotinplacerestore <my_restore_name> \
--snapshot <namespace/snapshot_to_restore> \
-n <application_namespace>
```

Verifique o status de uma operação de restauração

Você pode usar a linha de comando para verificar o status de uma operação de restauração que está em andamento, concluiu ou falhou.

Passos

1. Use o seguinte comando para recuperar o status da operação de restauração, substituindo valores em brackes por informações do seu ambiente:

```
kubectl get backuprestore -n <namespace_name> <my_restore_cr_name> -o
jsonpath='{.status}'
```

Utilize as configurações avançadas de restauração do Trident Protect.

Você pode personalizar operações de restauração usando configurações avançadas, como anotações, configurações de namespace e opções de armazenamento para atender aos seus requisitos específicos.

Anotações e rótulos de namespace durante operações de restauração e failover

Durante as operações de restauração e failover, rótulos e anotações no namespace de destino são feitos para corresponder aos rótulos e anotações no namespace de origem. Rótulos ou anotações do namespace de origem que não existem no namespace de destino são adicionados, e quaisquer rótulos ou anotações que já existem são sobrescritos para corresponder ao valor do namespace de origem. Rótulos ou anotações que existem apenas no namespace de destino permanecem inalterados.

OBSERVAÇÃO	Se você usa o Red Hat OpenShift, é importante observar o papel crítico das anotações de namespace em ambientes OpenShift. As anotações de namespace garantem que os pods restaurados cumpram as permissões e configurações de segurança apropriadas definidas pelas restrições de contexto de segurança (SCCs) do OpenShift e possam acessar volumes sem problemas de permissão. Para mais informações, consulte o"Documentação de restrições de contexto de segurança OpenShift" .

Você pode impedir que anotações específicas no namespace de destino sejam sobrescritas definindo a variável de ambiente do Kubernetes `RESTORE_SKIP_NAMESPACE_ANNOTATIONS` antes de executar a operação de restauração ou failover. Por exemplo:

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect \
  --set-string
restoreSkipNamespaceAnnotations="{<annotation_key_to_skip_1>,<annotation_k
ey_to_skip_2>}" \
  --reuse-values
```

OBSERVAÇÃO

Ao executar uma operação de restauração ou failover, quaisquer anotações e rótulos de namespace especificados em `restoreSkipNamespaceAnnotations` e `restoreSkipNamespaceLabels` são excluídos da operação de restauração ou failover. Certifique-se de que essas configurações sejam definidas durante a instalação inicial do Helm. Para saber mais, consulte "Configure as definições adicionais do gráfico de navegação do Trident Protect.".

Se você instalou o aplicativo de origem usando o Helm com o `--create-namespace` bandeira, tratamento especial é dado ao `name` Legenda da etiqueta. Durante o processo de restauração ou failover, o Trident Protect copia esse rótulo para o namespace de destino, mas atualiza o valor para o valor do namespace de destino se o valor da origem corresponder ao namespace de origem. Se esse valor não corresponder ao namespace de origem, ele será copiado para o namespace de destino sem alterações.

Exemplo

O exemplo a seguir apresenta um namespace de origem e destino, cada um com anotações e rótulos diferentes. Você pode ver o estado do namespace de destino antes e depois da operação e como as anotações e rótulos são combinados ou substituídos no namespace de destino.

Antes da operação de restauração ou failover

A tabela a seguir ilustra o estado dos namespaces de origem e destino de exemplo antes da operação de restauração ou failover:

Namespace	Anotações	Etiquetas
Namespace ns-1 (fonte)	• annotation.one/key: "updatedvalue" • annotation.two/key: "true"	• ambiente de produção • conformidade hipaa • nome: ns-1
Namespace ns-2 (destino)	• annotation.one/key: "true" (verdadeiro) • annotation.three/key: "false"	• banco de dados

Após a operação de restauração

A tabela a seguir ilustra o estado do namespace de destino de exemplo após a operação de restauração ou failover. Algumas chaves foram adicionadas, algumas foram sobrescritas e o `name` rótulo foi atualizado para corresponder ao namespace de destino:

Namespace	Anotações	Etiquetas
Namespace ns-2 (destino)	• annotation.one/key: "updatedvalue" • annotation.two/key: "true" • annotation.three/key: "false"	• nome: ns-2 • conformidade hipaa • ambiente de produção • banco de dados

Campos suportados

Esta seção descreve campos adicionais disponíveis para operações de restauração.

Mapeamento de classes de armazenamento

O `spec.storageClassMapping` atributo define um mapeamento de uma classe de armazenamento presente no aplicativo de origem para uma nova classe de armazenamento no cluster de destino. Você pode usar isso ao migrar aplicativos entre clusters com diferentes classes de armazenamento ou ao alterar o backend de armazenamento para operações de BackupRestore.

Exemplo:

```
storageClassMapping:
  - destination: "destinationStorageClass1"
    source: "sourceStorageClass1"
  - destination: "destinationStorageClass2"
    source: "sourceStorageClass2"
```

Anotações suportadas

Esta seção lista as anotações suportadas para configurar diversos comportamentos no sistema. Se uma anotação não for definida explicitamente pelo usuário, o sistema usará o valor padrão.

Anotação	Tipo	Descrição	Valor padrão
protect.trident.netapp.io/data-mover-timeout-sec	cadeia de carateres	O tempo máximo (em segundos) permitido para a operação do movimentador de dados ser interrompida.	"300"
protect.trident.netapp.io/kopia-content-cache-size-limit-mb	cadeia de carateres	O limite máximo de tamanho (em megabytes) para o cache de conteúdo do Kopia.	"1000"
protect.trident.netapp.io/pvc-bind-timeout-sec	cadeia de carateres	Tempo máximo (em segundos) de espera para que quaisquer PersistentVolumeClaims (PVCs) recém-criados cheguem ao `Bound` fase anterior à falha das operações. Aplica-se a todos os tipos de restauração de CR (BackupRestore, BackupInplaceRestore, SnapshotRestore, SnapshotInplaceRestore). Use um valor mais alto se o seu sistema de armazenamento ou cluster frequentemente exigir mais tempo.	"1200" (20 minutos)

Anotação	Tipo	Descrição	Valor padrão
protect.trident.netapp.io/retain-mac-address	cadeia de carateres	Para VMs do KubeVirt, defina como `"true"` para manter o endereço MAC da primeira interface de rede na restauração, por exemplo, quando uma licença de aplicativo estiver vinculada a ele. Aplica-se a todos os tipos de CR de restauração e ao AppMirrorRelationship (defina isso antes de promover o relacionamento). Por padrão, o Trident Protect remove esse endereço MAC para evitar conflitos. Se a VM de origem ainda estiver em execução na mesma rede, desligue-a ou primeiro restaure-a em uma rede isolada para evitar um endereço MAC duplicado.	Não configurado (o endereço MAC foi removido)

Modifique os recursos à medida que os restaura usando recursos personalizados

As transformações de recursos permitem que você modifique um recurso durante a restauração. Isso é útil quando a versão restaurada precisa ser diferente da original, por exemplo, ao atualizar um nome de host em um ConfigMap ao restaurar um aplicativo para um cluster diferente.

Como funciona a modificação de recursos

O campo `transformations` em `SnapshotRestore`, `BackupRestore`, `AppMirrorRelationship` e outros recursos de restauração permite que você modifique recursos do Kubernetes durante o processo de restauração. Isso é útil quando o aplicativo restaurado precisa ser configurado de forma diferente do original, por exemplo, alterando nomes de host, URLs de registro, limites de recursos ou variáveis de ambiente.

As transformações de recursos utilizam "RFC 6902 JSON Patch"operações e "RFC 6901 JSON Pointer"caminhos para direcionar e modificar campos específicos dentro dos recursos do Kubernetes.

A seguir, é apresentada a estrutura básica de um objeto de restauração que contém transformações de recursos:

```
apiVersion: protect.trident.netapp.io/v1
kind: SnapshotRestore
metadata:
  name: my-restore
  namespace: target-namespace
spec:
  appVaultRef: my-vault
  appArchivePath: /path/to/snapshot
  namespaceMapping:
    - source: source-ns
      destination: target-ns
  transformations:
    - resource:
        kind: ConfigMap       # Required: resource kind
        group: ""             # Optional: API group (empty for core
```

```
resources)
        version: ""           # Optional: API version
        name: ""              # Optional: specific resource name
      operations:
        - op: replace         # Operation type
          path: "/data/key"   # JSON Pointer path
          value: "new-value"  # New value (for add/replace/test)
```

Recursos suportados

Você pode usar transformações de recursos com recursos que correspondam aos seguintes critérios:

- **tipo** (obrigatório): o tipo de recurso do Kubernetes (por exemplo, `ConfigMap`, `Deployment`, `Pod`)
- **grupo** (opcional): o grupo da API (por exemplo, `apps`, `route.openshift.io`) — omita para recursos principais
- **versão** (opcional): a versão da API (por exemplo, `v1`, `v1beta1`)
- **nome** (opcional): aplicar somente a um recurso específico pelo nome

OBSERVAÇÃO Modificar PersistentVolumeClaims e namespaces não é suportado atualmente.

Operações suportadas

Você pode usar as seguintes operações para modificar recursos: `add`, `copy`, `move`, `remove`, `replace` e `test`.

Adicionar um valor a um recurso

Use a `add`operação para adicionar um novo campo ou valor no caminho especificado. Você pode adicionar dados a objetos ou arrays.

Use um CR

O exemplo a seguir adiciona um seletor de nó a um `Deployment` recurso:

```
transformations:
  - resource:
      kind: Deployment
    operations:
      - op: add
        path: "/spec/template/spec/nodeSelector"
        value:
          "topology.kubernetes.io/zone": "us-east-1a"
          disktype: "ssd"
```

Use a CLI

Utilize o seguinte comando para executar essa transformação usando a linha de comando:

```
tridentctl-protect create snapshotrestore <restore_name> \
  --snapshot <namespace/snapshot-name> \
  --namespace-mapping <source-ns>:<dest-ns> \
  --transformation
'apps,v1,Deployment:add{"path":"/spec/template/spec/nodeSelector","valu
e":{"topology.kubernetes.io/zone":"us-east-1a","disktype":"ssd"}}'
```

Copiar um valor dentro de um recurso

Use a operação `copy` para copiar um valor de um caminho para outro dentro do mesmo recurso. A origem não é alterada.

Use um CR

O exemplo a seguir duplica uma chave de dados para um `ConfigMap` objeto:

```
transformations:
  - resource:
      kind: ConfigMap
    operations:
      - op: copy
        from: "/data/source-key"
        path: "/data/backup-key"
```

Use a CLI

Utilize o seguinte comando para executar essa transformação usando a linha de comando:

```
tridentctl-protect create snapshotrestore <restore_name> \
  --snapshot <namespace/snapshot-name> \
  --namespace-mapping <source-ns>:<dest-ns> \
  --transformation ',v1,ConfigMap:copy{"from":"/data/source-
key","path":"/data/backup-key"}'
```

Mover um valor dentro de um recurso

Use a `move`operação para mover um valor de um caminho para outro dentro do mesmo recurso. A origem é removida e o valor é colocado no destino.

Use um CR

O exemplo a seguir renomeia uma chave de dados para um `ConfigMap` objeto:

```
transformations:
  - resource:
      kind: ConfigMap
    operations:
      - op: move
        from: "/data/OLD_KEY"
        path: "/data/NEW_KEY"
```

Use a CLI

Utilize o seguinte comando para executar essa transformação usando a linha de comando:

```
tridentctl-protect create snapshotrestore <restore_name> \
  --snapshot <namespace/snapshot-name> \
  --namespace-mapping <source-ns>:<dest-ns> \
  --transformation
',v1,ConfigMap:move{"from":"/data/OLD_KEY","path":"/data/NEW_KEY"}'
```

Remover um valor de um recurso

Utilize a `remove` operação para remover um campo ou valor no caminho especificado.

Use um CR

O exemplo a seguir remove uma anotação de um recurso `ConfigMap`:

```
transformations:
  - resource:
      kind: ConfigMap
    operations:
      - op: remove
        path: "/metadata/annotations/kubectl.kubernetes.io~1last-
applied-configuration"
```

OBSERVAÇÃO	No caminho do exemplo acima, `~1` é a sequência de escape do ponteiro JSON para `/`.

Use a CLI

Utilize o seguinte comando para executar essa transformação usando a linha de comando:

```
tridentctl-protect create snapshotrestore <restore_name> \
  --snapshot <namespace/snapshot-name> \
  --namespace-mapping <source-ns>:<dest-ns> \
```

```
  --transformation
',v1,ConfigMap:remove{"path":"/metadata/annotations/kubectl.kubernetes.
io~1last-applied-configuration"}'
```

Substituir um valor dentro de um recurso

Use a operação `replace` para substituir um valor existente em um recurso no caminho especificado. O caminho JSON já deve existir.

Use um CR

O exemplo a seguir altera o nome do host de um `Route` objeto:

```
transformations:
  - resource:
      kind: Route
      group: route.openshift.io
    operations:
      - op: replace
        path: "/spec/host"
        value: "prod.example.com"
```

Use a CLI

Utilize o seguinte comando para executar essa transformação usando a linha de comando:

```
tridentctl-protect create snapshotrestore <restore_name> \
  --snapshot <namespace/snapshot-name> \
  --namespace-mapping <source-ns>:<dest-ns> \
  --transformation
'route.openshift.io,v1,Route:replace{"path":"/spec/host","value":"prod.
example.com"}'
```

Teste a modificação do recurso

Use a operação `test` para verificar se o valor em um caminho corresponde ao valor esperado antes que as outras operações sejam aplicadas. Se o teste falhar, toda a modificação será revertida.

Use um CR

O exemplo a seguir atualiza `database-host` somente se `environment` estiver definido como `staging`:

```
transformations:
```

```
  - resource:
      kind: ConfigMap
    operations:
      - op: test
        path: "/data/environment"
        value: "staging"
      - op: replace
        path: "/data/database-host"
        value: "prod-db.example.com"
```

Use a CLI

Utilize o seguinte comando para executar essa transformação usando a linha de comando:

```
tridentctl-protect create snapshotrestore <restore_name> \
  --snapshot <namespace/snapshot-name> \
  --namespace-mapping <source-ns>:<dest-ns> \
  --transformation
',v1,ConfigMap:test{"path":"/data/environment","value":"staging"},repla
ce{"path":"/data/database-host","value":"prod-db.example.com"}'
```

Replique aplicações com o NetApp SnapMirror

Configure e gerencie a replicação de aplicativos usando o Trident Protect

Com o Trident Protect, você pode usar os recursos de replicação assíncrona da tecnologia NetApp SnapMirror para replicar dados e alterações de aplicativos de um backend de armazenamento para outro, no mesmo cluster ou entre clusters diferentes.

Anotações e rótulos de namespace durante operações de restauração e failover

Durante as operações de restauração e failover, rótulos e anotações no namespace de destino são feitos para corresponder aos rótulos e anotações no namespace de origem. Rótulos ou anotações do namespace de origem que não existem no namespace de destino são adicionados, e quaisquer rótulos ou anotações que já existem são sobrescritos para corresponder ao valor do namespace de origem. Rótulos ou anotações que existem apenas no namespace de destino permanecem inalterados.

OBSERVAÇÃO	Se você usa o Red Hat OpenShift, é importante observar o papel crítico das anotações de namespace em ambientes OpenShift. As anotações de namespace garantem que os pods restaurados cumpram as permissões e configurações de segurança apropriadas definidas pelas restrições de contexto de segurança (SCCs) do OpenShift e possam acessar volumes sem problemas de permissão. Para mais informações, consulte o"Documentação de restrições de contexto de segurança OpenShift" .

Você pode impedir que anotações específicas no namespace de destino sejam sobrescritas definindo a variável de ambiente do Kubernetes `RESTORE_SKIP_NAMESPACE_ANNOTATIONS` antes de executar a operação de restauração ou failover. Por exemplo:

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect \
  --set-string
restoreSkipNamespaceAnnotations="{<annotation_key_to_skip_1>,<annotation_k
ey_to_skip_2>}" \
  --reuse-values
```

OBSERVAÇÃO

Ao executar uma operação de restauração ou failover, quaisquer anotações e rótulos de namespace especificados em `restoreSkipNamespaceAnnotations` e `restoreSkipNamespaceLabels` são excluídos da operação de restauração ou failover. Certifique-se de que essas configurações sejam definidas durante a instalação inicial do Helm. Para saber mais, consulte "Configure as definições adicionais do gráfico de navegação do Trident Protect.".

Se você instalou o aplicativo de origem usando o Helm com o `--create-namespace` bandeira, tratamento especial é dado ao `name` Legenda da etiqueta. Durante o processo de restauração ou failover, o Trident Protect copia esse rótulo para o namespace de destino, mas atualiza o valor para o valor do namespace de destino se o valor da origem corresponder ao namespace de origem. Se esse valor não corresponder ao namespace de origem, ele será copiado para o namespace de destino sem alterações.

Exemplo

O exemplo a seguir apresenta um namespace de origem e destino, cada um com anotações e rótulos diferentes. Você pode ver o estado do namespace de destino antes e depois da operação e como as anotações e rótulos são combinados ou substituídos no namespace de destino.

Antes da operação de restauração ou failover

A tabela a seguir ilustra o estado dos namespaces de origem e destino de exemplo antes da operação de restauração ou failover:

Namespace	Anotações	Etiquetas
Namespace ns-1 (fonte)	• annotation.one/key: "updatedvalue" • annotation.two/key: "true"	• ambiente de produção • conformidade hipaa • nome: ns-1
Namespace ns-2 (destino)	• annotation.one/key: "true" (verdadeiro) • annotation.three/key: "false"	• banco de dados

Após a operação de restauração

A tabela a seguir ilustra o estado do namespace de destino de exemplo após a operação de restauração ou failover. Algumas chaves foram adicionadas, algumas foram sobrescritas e o `name` rótulo foi atualizado para corresponder ao namespace de destino:

Namespace	Anotações	Etiquetas
Namespace ns-2 (destino)	• annotation.one/key: "updatedvalue" • annotation.two/key: "true" • annotation.three/key: "false"	• nome: ns-2 • conformidade hipaa • ambiente de produção • banco de dados

OBSERVAÇÃO	Você pode configurar o Trident Protect para congelar e descongelar sistemas de arquivos durante operações de proteção de dados. "Saiba mais sobre como configurar o congelamento do sistema de arquivos com o Trident Protect.".

Ganchos de execução durante operações de failover e reverso

Ao usar o relacionamento do AppMirror para proteger seu aplicativo, há comportamentos específicos relacionados aos ganchos de execução dos quais você deve estar ciente durante operações de failover e reverso.

- Durante o failover, os ganchos de execução são copiados automaticamente do cluster de origem para o cluster de destino. Não é necessário recriá-los manualmente. Após o failover, os ganchos de execução permanecem presentes no aplicativo e serão executados durante quaisquer ações relevantes.
- Durante a sincronização reversa ou ressincronização reversa, quaisquer ganchos de execução existentes no aplicativo são removidos. Quando o aplicativo de origem se torna o aplicativo de destino, esses ganchos de execução não são válidos e são excluídos para impedir sua execução.

Para saber mais sobre ganchos de execução, consulte"Gerenciar os ganchos de execução do Trident Protect" .

Configure uma relação de replicação

A configuração de uma relação de replicação envolve o seguinte:

- Escolher a frequência com que o Trident Protect deve tirar um snapshot do aplicativo (que inclui os recursos do Kubernetes do aplicativo, bem como os snapshots de volume para cada um dos volumes do aplicativo).
- Escolha do cronograma de replicação (inclui recursos do Kubernetes e dados de volume persistente)
- Definir o tempo para a captura instantânea

Passos

1. No cluster de origem, crie um AppVault para o aplicativo de origem. Dependendo do seu fornecedor de storage, modifique um exemplo no "Recursos personalizados do AppVault" para se adequar ao seu ambiente:

   Crie um AppVault usando um CR

   a. Crie o arquivo de recurso personalizado (CR) e nomeie-o (por exemplo, `trident-protect-appvault-primary-source.yaml`).

   b. Configure os seguintes atributos:

      - **metadata.name**: (*required*) o nome do recurso personalizado do AppVault. Anote o nome que você escolher, porque outros arquivos CR necessários para uma relação de replicação referem-se a esse valor.

- **spec.providerConfig**: (*required*) armazena a configuração necessária para acessar o AppVault usando o provedor especificado. Escolha um bucketName e quaisquer outros detalhes necessários para o seu provedor. Anote os valores que você escolher, porque outros arquivos CR necessários para uma relação de replicação se referem a esses valores. "Recursos personalizados do AppVault"Consulte para obter exemplos de AppVault CRS com outros provedores.
- **spec.providerCredentials**: (*required*) armazena referências a qualquer credencial necessária para acessar o AppVault usando o provedor especificado.
  - **spec.providerCredentials.valueFromSecret**: (*required*) indica que o valor da credencial deve vir de um segredo.
    - **Key**: (*required*) a chave válida do segredo para selecionar.
    - **Name**: (*required*) Nome do segredo que contém o valor deste campo. Deve estar no mesmo namespace.
  - **spec.providerCredentials.secretAccessKey**: (*required*) a chave de acesso usada para acessar o provedor. O **nome** deve corresponder a **spec.providerCredentials.valueFromSecret.name**.
- **spec.providerType**: (*required*) determina o que fornece o backup; por exemplo, NetApp ONTAP S3, S3 genérico, Google Cloud ou Microsoft Azure. Valores possíveis:
  - aws
  - azure
  - gcp
  - generic-s3
  - ONTAP-s3
  - StorageGRID-s3

c. Depois de preencher o `trident-protect-appvault-primary-source.yaml` ficheiro com os valores corretos, aplique o CR:

```
kubectl apply -f trident-protect-appvault-primary-source.yaml -n
trident-protect
```

Crie um AppVault usando a CLI

a. Crie o AppVault, substituindo valores entre parênteses por informações do seu ambiente:

```
tridentctl-protect create vault Azure <vault-name> --account
<account-name> --bucket <bucket-name> --secret <secret-name> -n
trident-protect
```

2. No cluster de origem, crie a aplicação de origem CR:

Crie o aplicativo de origem usando um CR

a. Crie o arquivo de recurso personalizado (CR) e nomeie-o (por exemplo, `trident-protect-app-source.yaml`).

b. Configure os seguintes atributos:

- **metadata.name**: (*required*) o nome do recurso personalizado do aplicativo. Anote o nome que você escolher, porque outros arquivos CR necessários para uma relação de replicação referem-se a esse valor.
- **spec.includedNamespaces**: (*required*) um array de namespaces e rótulos associados. Use nomes de namespace e, opcionalmente, restrinja o escopo dos namespaces com rótulos para especificar recursos que existem nos namespaces listados aqui. O namespace da aplicação deve fazer parte desse array.

  Exemplo YAML:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: Application
metadata:
  name: my-app-name
  namespace: my-app-namespace
spec:
  includedNamespaces:
    - namespace: my-app-namespace
      labelSelector: {}
```

c. Depois de preencher o `trident-protect-app-source.yaml` ficheiro com os valores corretos, aplique o CR:

```
kubectl apply -f trident-protect-app-source.yaml -n my-app-
namespace
```

Crie o aplicativo de origem usando a CLI

a. Crie o aplicativo de origem. Por exemplo:

```
tridentctl-protect create app <my-app-name> --namespaces
<namespaces-to-be-included> -n <my-app-namespace>
```

3. Opcionalmente, no cluster de origem, tire um snapshot do aplicativo de origem. Este instantâneo é usado como base para a aplicação no cluster de destino. Se você pular esta etapa, precisará aguardar a próxima captura de instantâneo agendada para obter um instantâneo recente. Para criar um instantâneo sob demanda, consulte "Crie um snapshot sob demanda".
4. No cluster de origem, crie a solicitação de configuração (CR) para o agendamento de replicação:

OBSERVAÇÃO	Além do cronograma fornecido abaixo, recomenda-se criar um cronograma de snapshots diários separado, com um período de retenção de 7 dias, para manter um snapshot comum entre clusters ONTAP pareados. Isso garante que os snapshots fiquem disponíveis por até 7 dias, mas o período de retenção

pode ser personalizado de acordo com as necessidades do usuário.

Em caso de failover, o sistema pode usar esses snapshots por até 7 dias para operações reversas. Essa abordagem torna o processo de reversão mais rápido e eficiente, pois apenas as alterações feitas desde o último snapshot serão transferidas, e não todos os dados.

Se um cronograma existente para o aplicativo já atender aos requisitos de retenção desejados, nenhum cronograma adicional será necessário.

Crie o cronograma de replicação usando uma CR (Código de Referência).

a. Crie um agendamento de replicação para o aplicativo de origem:

   i. Crie o arquivo de recurso personalizado (CR) e nomeie-o (por exemplo, `trident-protect-schedule.yaml`).

   ii. Configure os seguintes atributos:

   - **metadata.name**: (*required*) o nome do recurso personalizado de agendamento.
   - **spec.appVaultRef**: (*Obrigatório*) Este valor deve corresponder ao campo metadata.name do AppVault para o aplicativo de origem.
   - **spec.applicationRef**: (*Obrigatório*) Este valor deve corresponder ao campo metadata.name do CR da aplicação de origem.
   - **Spec.backupRetention**: (*required*) este campo é obrigatório e o valor deve ser definido como 0.
   - **Spec.enabled**: Deve ser definido como true.
   - **spec.granularity**: tem de estar definido para `Custom`.
   - **Spec.recurrenceRule**: Defina uma data de início no horário UTC e um intervalo de recorrência.
   - **Spec.snapshotRetention**: Deve ser definido como 2.

     Exemplo YAML:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: Schedule
metadata:
  name: appmirror-schedule
  namespace: my-app-namespace
spec:
  appVaultRef: my-appvault-name
  applicationRef: my-app-name
  backupRetention: "0"
  enabled: true
  granularity: Custom
  recurrenceRule: |-
    DTSTART:20220101T000200Z
    RRULE:FREQ=MINUTELY;INTERVAL=5
```

```
  snapshotRetention: "2"
```

i. Depois de preencher o `trident-protect-schedule.yaml` ficheiro com os valores corretos, aplique o CR:

```
kubectl apply -f trident-protect-schedule.yaml -n my-app-namespace
```

Crie o agendamento de replicação usando a CLI

a. Crie o cronograma de replicação, substituindo os valores entre colchetes pelas informações do seu ambiente:

```
tridentctl-protect create schedule --name appmirror-schedule
--app <my_app_name> --appvault <my_app_vault> --granularity
Custom --recurrence-rule <rule> --snapshot-retention
<snapshot_retention_count> -n <my_app_namespace>
```

Exemplo:

```
tridentctl-protect create schedule --name appmirror-schedule
--app <my_app_name> --appvault <my_app_vault> --granularity
Custom --recurrence-rule  "DTSTART:20220101T000200Z
\nRRULE:FREQ=MINUTELY;INTERVAL=5" --snapshot-retention 2 -n
<my_app_namespace>
```

5. No cluster de destino, crie um aplicativo de origem AppVault CR idêntico ao AppVault CR aplicado no cluster de origem e nomeie-o (por exemplo, `trident-protect-appvault-primary-destination.yaml`).
6. Aplicar o CR:

```
kubectl apply -f trident-protect-appvault-primary-destination.yaml -n trident-protect
```

7. Crie um AppVault CR de destino para o aplicativo de destino no cluster de destino. Dependendo do seu fornecedor de storage, modifique um exemplo no "Recursos personalizados do AppVault" para se adequar ao seu ambiente:

a. Crie o arquivo de recurso personalizado (CR) e nomeie-o (por exemplo, `trident-protect-appvault-secondary-destination.yaml`).

b. Configure os seguintes atributos:

- **metadata.name**: (*required*) o nome do recurso personalizado do AppVault. Anote o nome que você escolher, porque outros arquivos CR necessários para uma relação de replicação referem-se

a esse valor.

- **spec.providerConfig**: (*required*) armazena a configuração necessária para acessar o AppVault usando o provedor especificado. Escolha um `bucketName` e quaisquer outros detalhes necessários para o seu provedor. Anote os valores que você escolher, porque outros arquivos CR necessários para uma relação de replicação se referem a esses valores. "Recursos personalizados do AppVault"Consulte para obter exemplos de AppVault CRS com outros provedores.
- **spec.providerCredentials**: (*required*) armazena referências a qualquer credencial necessária para acessar o AppVault usando o provedor especificado.
  - **spec.providerCredentials.valueFromSecret**: (*required*) indica que o valor da credencial deve vir de um segredo.
    - **Key**: (*required*) a chave válida do segredo para selecionar.
    - **Name**: (*required*) Nome do segredo que contém o valor deste campo. Deve estar no mesmo namespace.
  - **spec.providerCredentials.secretAccessKey**: (*required*) a chave de acesso usada para acessar o provedor. O **nome** deve corresponder a **spec.providerCredentials.valueFromSecret.name**.
- **spec.providerType**: (*required*) determina o que fornece o backup; por exemplo, NetApp ONTAP S3, S3 genérico, Google Cloud ou Microsoft Azure. Valores possíveis:
  - aws
  - azure
  - gcp
  - generic-s3
  - ONTAP-s3
  - StorageGRID-s3

c. Depois de preencher o `trident-protect-appvault-secondary-destination.yaml` ficheiro com os valores corretos, aplique o CR:

```
kubectl apply -f trident-protect-appvault-secondary-destination.yaml
-n trident-protect
```

8. No cluster de destino, crie um arquivo CR AppMirrorRelationship.

OBSERVAÇÃO	Ao usar uma solicitação de alteração (CR), crie manualmente o namespace de destino antes de aplicar a CR. O Trident Protect cria namespaces automaticamente apenas quando se utiliza a CLI (linha de comando).

Crie um AppMirrorRelationship usando um CR

a. Crie o arquivo de recurso personalizado (CR) e nomeie-o (por exemplo, `trident-protect-relationship.yaml`).

b. Configure os seguintes atributos:

- **metadata.name:** (obrigatório) o nome do recurso personalizado AppMirrorRelationship.

- **metadata.annotations.protect.trident.netapp.io/retain-mac-address**: (*Opcional*) Defina como `"true"` para reter o endereço MAC das VMs do KubeVirt no cluster de destino. Consulte "Utilize as configurações avançadas de restauração do Trident Protect.".
- **spec.destinationAppVaultRef**: (*required*) esse valor deve corresponder ao nome do AppVault para o aplicativo de destino no cluster de destino.
- **spec.namespaceMapping**: (*required*) os namespaces de destino e origem devem corresponder ao namespace de aplicativo definido no respetivo CR de aplicação.
- **Spec.sourceAppVaultRef**: (*required*) este valor deve corresponder ao nome do AppVault para o aplicativo de origem.
- **Spec.sourceApplicationName**: (*required*) esse valor deve corresponder ao nome do aplicativo de origem definido no CR do aplicativo de origem.
- **spec.sourceApplicationUID**: (Obrigatório) Este valor deve corresponder ao UID do aplicativo de origem que você definiu no CR do aplicativo de origem.
- **spec.storageClassName**: (Opcional) Escolha o nome de uma classe de armazenamento válida no cluster. A classe de armazenamento deve ser vinculada a uma VM de armazenamento ONTAP que esteja emparelhada com o ambiente de origem. Caso a classe de armazenamento não seja especificada, a classe de armazenamento padrão do cluster será utilizada por padrão.
- **spec.storageClassMapping**: (*Opcional*) Mapeia classes de armazenamento de origem para classes de armazenamento de destino para PVCs individuais dentro do mesmo AMR. Cada entrada requer um `source` e `destination` nome de classe de armazenamento. Quando definido, `storageClassMapping` substitui `storageClassName` completamente; PVCs sem uma entrada de origem correspondente usam a classe de armazenamento padrão do cluster, não `storageClassName`.

  Ao utilizar `storageClassMapping` em um AppMirrorRelationship CR, esteja ciente das seguintes restrições:

  - `storageClassMapping` substitui `storageClassName` completamente. Quando ambos os campos estão definidos, `storageClassName` é ignorado.
  - Os PVCs com classes de armazenamento de origem que não possuem entrada correspondente em `storageClassMapping` usam a classe de armazenamento padrão do cluster, não em `storageClassName`.
  - Cada entrada requer um valor `source` e um valor `destination`.
  - Alterar a classe de armazenamento de destino pode forçar a exclusão e recriação dos PVCs de destino existentes.
- **Spec.recurrenceRule**: Defina uma data de início no horário UTC e um intervalo de recorrência.

  Exemplo YAML:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: AppMirrorRelationship
metadata:
  name: amr-16061e80-1b05-4e80-9d26-d326dc1953d8
  namespace: my-app-namespace
```

```
spec:
  desiredState: Established
  destinationAppVaultRef: generic-s3-trident-protect-dst-bucket-
8fe0b902-f369-4317-93d1-ad7f2edc02b5
  namespaceMapping:
    - destination: my-app-namespace
      source: my-app-namespace
  recurrenceRule: |-
    DTSTART:20220101T000200Z
    RRULE:FREQ=MINUTELY;INTERVAL=5
  sourceAppVaultRef: generic-s3-trident-protect-src-bucket-
b643cc50-0429-4ad5-971f-ac4a83621922
  sourceApplicationName: my-app-name
  sourceApplicationUID: 7498d32c-328e-4ddd-9029-122540866aeb
  storageClassName: sc-vsim-2
```

c. Depois de preencher o `trident-protect-relationship.yaml` ficheiro com os valores corretos, aplique o CR:

```
kubectl apply -f trident-protect-relationship.yaml -n my-app-
namespace
```

Crie um AppMirrorRelationship usando a CLI

a. Crie e aplique o objeto AppMirrorRelationship, substituindo os valores entre colchetes pelas informações do seu ambiente:

```
tridentctl-protect create appmirrorrelationship
<name_of_appmirorrelationship> --destination-app-vault
<my_vault_name> --source-app-vault <my_vault_name> --recurrence
-rule <rule> --namespace-mapping <ns_mapping> --source-app-id
<source_app_UID> --source-app <my_source_app_name> --storage
-class <storage_class_name> --storageclass-mapping
<src_class:dst_class> -n <application_namespace>
```

OBSERVAÇÃO	Use `--storageclass-mapping` para mapear classes de armazenamento de origem individuais para classes de armazenamento de destino. Especifique um ou mais pares `source:destination` separados por vírgulas (por exemplo, `"class-a:class-b,class-c:class-d"`). Use `*:destination-class` como um caractere curinga para mapear todos os PVCs não mapeados para uma única classe de armazenamento de destino. Quando `--storageclass-mapping` é definido, `--storage-class` é ignorado.

Exemplo usando uma única classe de armazenamento de destino (`--storage-class`:

```
tridentctl-protect create appmirrorrelationship my-amr
--destination-app-vault appvault2 --source-app-vault appvault1
--recurrence-rule
"DTSTART:20220101T000200Z\nRRULE:FREQ=MINUTELY;INTERVAL=5"
--source-app my-app --namespace-mapping "my-source-ns1:my-dest-
ns1,my-source-ns2:my-dest-ns2" --source-app-id 373f24c1-5769-
404c-93c3-5538af6ccc36 --storage-class my-storage-class -n my-
dest-ns1
```

9. (*Optional*) no cluster de destino, verifique o estado e o estado da relação de replicação:

```
kubectl get amr -n my-app-namespace <relationship name> -o=jsonpath
='{.status}' | jq
```

Failover para o cluster de destino

Com o Trident Protect, você pode realizar o failover de aplicativos replicados para um cluster de destino. Este procedimento interrompe a relação de replicação e coloca o aplicativo online no cluster de destino. O Trident Protect não interrompe o aplicativo no cluster de origem se ele estiver operacional.

Este procedimento faz failover para o snapshot mais recente. Para fazer failover para um snapshot anterior, consulte "Faça failover para um snapshot anterior usando o Trident Protect".

Passos

1. (*Opcional*) Para preservar o endereço MAC das VMs do KubeVirt, anote o AppMirrorRelationship antes de promovê-lo. Consulte "Utilize as configurações avançadas de restauração do Trident Protect.".

```
kubectl annotate appmirrorrelationship <relationship_name> -n
<application_namespace> protect.trident.netapp.io/retain-mac-address
="true"
```

2. No cluster de destino, edite o arquivo CR AppMirrorRelationship (por exemplo, `trident-protect-relationship.yaml`) e altere o valor de **spec.desiredState** para `Promoted`.
3. Salve o arquivo CR.
4. Aplicar o CR:

```
kubectl apply -f trident-protect-relationship.yaml -n my-app-namespace
```

5. (*Optional*) Crie todos os programas de proteção que você precisa no aplicativo com falha.
6. (*Optional*) Verifique o estado e o estado da relação de replicação:

```
kubectl get amr -n my-app-namespace <relationship name> -o=jsonpath
```

```
='{.status}' | jq
```

Ressincronizar uma relação de replicação com falha

A operação ressincronizada restabelece a relação de replicação. Depois de executar uma operação ressincronizada, o aplicativo de origem original se torna o aplicativo em execução e quaisquer alterações feitas no aplicativo em execução no cluster de destino serão descartadas.

O processo pára o aplicativo no cluster de destino antes de restabelecer a replicação.

IMPORTANTE	Todos os dados gravados na aplicação de destino durante o failover serão perdidos.

Passos

1. Opcional: No cluster de origem, crie um instantâneo do aplicativo de origem. Isso garante que as alterações mais recentes do cluster de origem sejam capturadas.
2. No cluster de destino, edite o arquivo CR AppMirrorRelationship (por exemplo, `trident-protect-relationship.yaml`) e altere o valor de spec.desiredState para `Established`.
3. Salve o arquivo CR.
4. Aplicar o CR:

```
kubectl apply -f trident-protect-relationship.yaml -n my-app-namespace
```

5. Se você criou quaisquer programações de proteção no cluster de destino para proteger o aplicativo com falha, remova-os. Quaisquer programações restantes causam falhas de snapshot de volume.

Ressincronização reversa de uma relação de replicação com falha

Quando você faz a ressincronização reversa de uma relação de replicação com falha, o aplicativo de destino se torna o aplicativo de origem e a origem se torna o destino. As alterações feitas na aplicação de destino durante o failover são mantidas.

Passos

1. No cluster de destino original, exclua o AppMirrorRelationship CR. Isso faz com que o destino se torne a fonte. Se houver planos de proteção restantes no novo cluster de destino, remova-os.
2. Configure uma relação de replicação aplicando os arquivos CR usados originalmente para configurar a relação com os clusters opostos.
3. Certifique-se de que o novo destino (cluster de origem original) esteja configurado com o AppVault CRS.
4. Configure uma relação de replicação no cluster oposto, configurando valores para a direção inversa.

Sentido de replicação da aplicação inversa

Ao inverter a direção da replicação, o Trident Protect move a aplicação para o backend de armazenamento de destino, enquanto continua a replicar de volta para o backend de armazenamento de origem original. O Trident Protect interrompe o aplicativo de origem e replica os dados para o destino antes de realizar o failover para o aplicativo de destino.

Nesta situação, você está trocando a origem e o destino.

Passos

1. No cluster de origem, crie um instantâneo de encerramento:

Crie um instantâneo de encerramento utilizando um CR

a. Desative as programações de políticas de proteção para o aplicativo de origem.

b. Criar um ficheiro ShutdownSnapshot CR:

i. Crie o arquivo de recurso personalizado (CR) e nomeie-o (por exemplo, `trident-protect-shutdownsnapshot.yaml`).

ii. Configure os seguintes atributos:

- **metadata.name**: (*required*) o nome do recurso personalizado.
- **Spec.AppVaultRef**: (*required*) este valor deve corresponder ao campo metadata.name do AppVault para o aplicativo de origem.
- **Spec.ApplicationRef**: (*required*) este valor deve corresponder ao campo metadata.name do arquivo CR da aplicação de origem.

Exemplo YAML:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: ShutdownSnapshot
metadata:
  name: replication-shutdown-snapshot-afc4c564-e700-4b72-86c3-
c08a5dbe844e
  namespace: my-app-namespace
spec:
  appVaultRef: generic-s3-trident-protect-src-bucket-04b6b4ec-
46a3-420a-b351-45795e1b5e34
  applicationRef: my-app-name
```

c. Depois de preencher o `trident-protect-shutdownsnapshot.yaml` ficheiro com os valores corretos, aplique o CR:

```
kubectl apply -f trident-protect-shutdownsnapshot.yaml -n my-app-
namespace
```

Crie um instantâneo de encerramento usando a CLI

a. Crie o instantâneo de encerramento, substituindo valores entre parênteses por informações do seu ambiente. Por exemplo:

```
tridentctl-protect create shutdownsnapshot <my_shutdown_snapshot>
--appvault <my_vault> --app <app_to_snapshot> -n
<application_namespace>
```

2. No cluster de origem, após a conclusão do instantâneo de encerramento, obtenha o status do instantâneo de encerramento:

```
kubectl get shutdownsnapshot -n my-app-namespace
<shutdown_snapshot_name> -o yaml
```

3. No cluster de origem, encontre o valor de **shutdownsnapshot.status.appArchivePath** usando o seguinte comando, e Registre a última parte do caminho do arquivo (também chamado de basename; isso será tudo após a última barra):

```
k get shutdownsnapshot -n my-app-namespace <shutdown_snapshot_name> -o
jsonpath='{.status.appArchivePath}'
```

4. Faça um failover do novo cluster de destino para o novo cluster de origem, com a seguinte alteração:

OBSERVAÇÃO	Na etapa 2 do procedimento de failover, inclua o `spec.promotedSnapshot` campo no arquivo AppMirrorRelationship CR e defina seu valor para o nome de base registrado na etapa 3 acima.

5. Execute as etapas de ressincronização reversa no Ressincronização reversa de uma relação de replicação com falha.
6. Ative programações de proteção no novo cluster de origem.

Resultado

As seguintes ações ocorrem devido à replicação reversa:

- Um snapshot é obtido dos recursos do Kubernetes do aplicativo de origem original.
- Os pods do aplicativo de origem original são interrompidos graciosamente ao excluir os recursos do Kubernetes do aplicativo (deixando PVCs e PVS no lugar).
- Depois que os pods são desativados, snapshots dos volumes do aplicativo são feitos e replicados.
- As relações do SnapMirror são quebradas, tornando os volumes de destino prontos para leitura/gravação.
- Os recursos do Kubernetes do aplicativo são restaurados a partir do snapshot de pré-encerramento, usando os dados de volume replicados após o desligamento do aplicativo de origem original.
- A replicação é restabelecida na direção inversa.

Falha de aplicativos para o cluster de origem original

Utilizando o Trident Protect, você pode realizar o "failback" após uma operação de failover seguindo a seguinte sequência de operações. Nesse fluxo de trabalho para restaurar a direção de replicação original, o Trident Protect replica (ressincroniza) quaisquer alterações do aplicativo de volta para o aplicativo de origem original antes de inverter a direção da replicação.

Esse processo começa a partir de um relacionamento que concluiu um failover para um destino e envolve as seguintes etapas:

- Comece com um estado com falha em excesso.

- Reverta a ressincronização da relação de replicação.

CUIDADO	Não execute uma operação de ressincronização normal, pois isso descartará os dados gravados no cluster de destino durante o procedimento de failover.

- Inverta a direção da replicação.

Passos

1. Execute os Ressincronização reversa de uma relação de replicação com falha passos.
2. Execute os Sentido de replicação da aplicação inversa passos.

Excluir uma relação de replicação

Você pode excluir um relacionamento de replicação a qualquer momento. Quando você exclui a relação de replicação do aplicativo, isso resulta em dois aplicativos separados sem relação entre eles.

Passos

1. No cluster de dessinização atual, exclua o AppMirrorRelationship CR:

```
kubectl delete -f trident-protect-relationship.yaml -n my-app-namespace
```

Faça failover para um snapshot anterior usando o Trident Protect

Use o Trident Protect para realizar o failover de uma aplicação replicada para um snapshot anterior, permitindo a recuperação a partir de um ponto no tempo anterior à ocorrência de danos ou comprometimento dos dados. Por padrão, o failover utiliza o snapshot transferido mais recentemente.

CUIDADO	Recorrer a um snapshot mais antigo faz com que o ONTAP SnapMirror exclua os snapshots no cluster de destino que são mais recentes do que o snapshot selecionado, porque eles não são mais válidos. Adicione a anotação neste procedimento somente se essa perda de dados for aceitável.

Antes de começar

- Um AppMirrorRelationship existe e está no `Established` estado. Para criar um, consulte "Configure uma relação de replicação". Para verificar o estado, execute `kubectl get amr -n <namespace> <relationship_name> -o=jsonpath='{.status}'`.
- Os clusters de origem e de destino usam backends de armazenamento ONTAP com replicação SnapMirror. O Azure NetApp Files e outros backends não ONTAP não são compatíveis. Consulte "Requisitos para replicação do SnapMirror".
- Existe um agendamento de snapshots (com `snapshotRetention` maior que 1) no aplicativo de origem.

Identifique o snapshot

Cada instantâneo tem um horário de conclusão, para que você possa escolher o ponto no tempo desejado.

Passos

1. No cluster de origem, liste os snapshots:

```
kubectl get snapshots -n my-app-namespace
```

2. Encontre o snapshot para o qual deseja fazer o failover. Use o `completionTimestamp`valor para selecionar o ponto no tempo.
3. Obtenha o caminho do arquivo desse snapshot:

```
kubectl get snapshots <snapshot_name> -n my-app-namespace -o jsonpath
='{.status.appArchivePath}'
```

4. Anote a última parte do caminho, ou seja, tudo o que vem depois da última barra. Essa parte é o *basename*. Por exemplo:

```
20231114173500_schedule-20231114173500_f8df8bd3-a727-4a14-b954-
9666aa618b55
```

OBSERVAÇÃO Use o basename, não o nome do recurso personalizado do snapshot.

Resultado

O nome base do snapshot selecionado é registrado.

Fazer failover para o snapshot selecionado

Especifique o snapshot a ser promovido e reconheça a perda de dados adicionando uma anotação.

Passos

1. No cluster de destino, adicione a anotação ao AppMirrorRelationship. Sem a anotação, um failover só poderá usar o snapshot transferido mais recentemente.

```
kubectl annotate appmirrorrelationship <relationship_name> -n my-app-
namespace protect.trident.netapp.io/allow-promotion-with-older-
snapshots="true"
```

2. Edite o arquivo CR AppMirrorRelationship (por exemplo, `trident-protect-relationship.yaml`):
   - Defina **spec.desiredState** para `Promoted`.
   - Defina **spec.promotedSnapshot** para o nome base que você registrou em Identifique o snapshot.

     Por exemplo:

```
apiVersion: protect.trident.netapp.io/v1
kind: AppMirrorRelationship
metadata:
  name: <relationship_name>
```

```
  namespace: <namespace>
  annotations:
    protect.trident.netapp.io/allow-promotion-with-older-snapshots:
"true"
spec:
  desiredState: Promoted
  promotedSnapshot: <basename>
```

3. Aplicar o CR:

```
kubectl apply -f trident-protect-relationship.yaml -n my-app-namespace
```

4. Verifique o status do relacionamento:

```
kubectl get amr -n my-app-namespace <relationship_name> -o=jsonpath
='{.status}' | jq
```

Resultado

O relacionamento passa para o `Promoted` estado, e o aplicativo é executado no cluster de destino com os dados do snapshot selecionado.

Se o relacionamento permanecer no `Promoting`estado e `status.error` mostrar `The snapshot provided is invalid or older than the most recently transferred snapshot`, a anotação está faltando ou não está definida como `true`. Adicione a anotação e tente novamente.

O que se segue?

Para retornar ao cluster original, execute um "ressincronização reversa" e, em seguida, "inverter a direção da replicação".

CUIDADO	Para retornar ao cluster original, não execute uma resincronização normal. Uma resincronização normal descarta a aplicação em failover e retorna o relacionamento ao snapshot de origem mais recente.

Informações relacionadas

- "Configure e gerencie a replicação de aplicativos usando o Trident Protect"
- "Failover para o cluster de destino"
- "Ressincronizar uma relação de replicação com falha"
- "Falha de aplicativos para o cluster de origem original"
- "Criar um cronograma de proteção de dados"

Migre aplicativos usando o Trident Protect.

Você pode migrar seus aplicativos entre clusters ou para diferentes classes de armazenamento restaurando dados de backup.

OBSERVAÇÃO	Quando você migra um aplicativo, todos os ganchos de execução configurados para o aplicativo são migrados com o aplicativo. Se um gancho de execução pós-restauração estiver presente, ele será executado automaticamente como parte da operação de restauração.

Operações de backup e restauração

Para executar operações de backup e restauração nos cenários a seguir, você pode automatizar tarefas específicas de backup e restauração.

Clonar para o mesmo cluster

Para clonar uma aplicação para o mesmo cluster, crie um snapshot ou backup e restaure os dados para o mesmo cluster.

Passos

1. Execute um dos seguintes procedimentos:
   a. "Criar um instantâneo".
   b. "Crie uma cópia de segurança".
2. No mesmo cluster, siga um destes procedimentos, dependendo se você criou um snapshot ou um backup:
   a. "Restaure seus dados a partir do snapshot".
   b. "Restaure seus dados a partir do backup".

Clone para cluster diferente

Para clonar um aplicativo para um cluster diferente (realizar uma clonagem entre clusters), crie um backup no cluster de origem e, em seguida, restaure o backup em um cluster diferente. Certifique-se de que o Trident Protect esteja instalado no cluster de destino.

OBSERVAÇÃO	É possível replicar um aplicativo entre clusters diferentes usando "Replicação SnapMirror"o .

Passos

1. "Crie uma cópia de segurança".
2. Verifique se o AppVault CR para o bucket de armazenamento de objetos que contém o backup foi configurado no cluster de destino.
3. No cluster de destino, "restaure seus dados a partir do backup".

Migrar aplicações de uma classe de storage para outra classe de storage

Você pode migrar aplicativos de uma classe de armazenamento para uma classe de armazenamento diferente restaurando um backup para a classe de armazenamento de destino.

Por exemplo (excluindo os segredos do CR de restauração):

```
apiVersion: protect.trident.netapp.io/v1
kind: SnapshotRestore
metadata:
```

```
  name: "${snapshotRestoreCRName}"
spec:
  appArchivePath: "${snapshotArchivePath}"
  appVaultRef: "${appVaultCRName}"
  namespaceMapping:
    - destination: "${destinationNamespace}"
      source: "${sourceNamespace}"
  storageClassMapping:
    - destination: "${destinationStorageClass}"
      source: "${sourceStorageClass}"
  resourceFilter:
    resourceMatchers:
      kind: Secret
      version: v1
    resourceSelectionCriteria: exclude
```

Restaure o instantâneo usando um CR

Passos

1. Crie o arquivo de recurso personalizado (CR) e nomeie-o `trident-protect-snapshot-restore-cr.yaml`.
2. No arquivo criado, configure os seguintes atributos:
   - **metadata.name**: (*required*) o nome deste recurso personalizado; escolha um nome único e sensível para o seu ambiente.
   - **Spec.appArchivePath**: O caminho dentro do AppVault onde o conteúdo do snapshot é armazenado. Você pode usar o seguinte comando para encontrar este caminho:

     ```
     kubectl get snapshots <my-snapshot-name> -n trident-protect -o jsonpath='{.status.appArchivePath}'
     ```

   - **Spec.appVaultRef**: (*required*) o nome do AppVault onde o conteúdo do instantâneo é armazenado.
   - **spec.namespaceMapping**: o mapeamento do namespace de origem da operação de restauração para o namespace de destino. Substitua `my-source-namespace` e `my-destination-namespace` por informações do seu ambiente.

     ```
     ---
     apiVersion: protect.trident.netapp.io/v1
     kind: SnapshotRestore
     metadata:
       name: my-cr-name
       namespace: trident-protect
     spec:
       appArchivePath: my-snapshot-path
     ```

```
  appVaultRef: appvault-name
  namespaceMapping: [{"source": "my-source-namespace",
"destination": "my-destination-namespace"}]
```

3. Opcionalmente, se você precisar selecionar apenas certos recursos do aplicativo para restaurar, adicione filtragem que inclua ou exclua recursos marcados com rótulos específicos:

   - **ResourceFilter.resourceSelectionCriteria**: (Necessário para filtragem) Use `include or exclude` para incluir ou excluir um recurso definido em resourceMatchers. Adicione os seguintes parâmetros resourceMatchers para definir os recursos a serem incluídos ou excluídos:
     - **ResourceFilter.resourceMatchers**: Uma matriz de resourceMatcher objetos. Se você definir vários elementos nesse array, eles corresponderão como uma OPERAÇÃO OU, e os campos dentro de cada elemento (grupo, tipo, versão) corresponderão como uma OPERAÇÃO E.
       - **ResourceMatchers[].group**: (*Optional*) Grupo do recurso a ser filtrado.
       - **ResourceMatchers[].kind**: (*Opcional*) tipo do recurso a ser filtrado.
       - **ResourceMatchers[].version**: (*Optional*) versão do recurso a ser filtrado.
       - **ResourceMatchers[].names**: (*Optional*) nomes no campo Kubernetes metadata.name do recurso a ser filtrado.
       - **ResourceMatchers[].namespaces**: (*Optional*) namespaces no campo Kubernetes metadata.name do recurso a ser filtrado.
       - **ResourceMatchers[].labelSelectors**: (*Optional*) string de seleção de etiquetas no campo Kubernetes metadata.name do recurso, conforme definido no "Documentação do Kubernetes". Por exemplo `"trident.netapp.io/os=linux":.`

         Por exemplo:

```
spec:
  resourceFilter:
    resourceSelectionCriteria: "include"
    resourceMatchers:
      - group: my-resource-group-1
        kind: my-resource-kind-1
        version: my-resource-version-1
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
      - group: my-resource-group-2
        kind: my-resource-kind-2
        version: my-resource-version-2
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
```

4. Depois de preencher o `trident-protect-snapshot-restore-cr.yaml` ficheiro com os valores corretos, aplique o CR:

```
kubectl apply -f trident-protect-snapshot-restore-cr.yaml
```

Restaure o instantâneo usando a CLI

Passos

1. Restaure o snapshot para um namespace diferente, substituindo valores entre parênteses por informações do seu ambiente.

   - O `snapshot` argumento usa um namespace e um nome instantâneo no formato `<namespace>/<name>`.
   - O `namespace-mapping` argumento usa namespaces separados por dois pontos para mapear namespaces de origem para os namespaces de destino corretos no formato `source1:dest1,source2:dest2`.

     Por exemplo:

```
tridentctl-protect create snapshotrestore <my_restore_name>
--snapshot <namespace/snapshot_to_restore> --namespace-mapping
<source_to_destination_namespace_mapping>
```

Proteja máquinas virtuais KubeVirt com Trident Protect

Trident Protect permite que você faça backup e restaure máquinas virtuais (VMs) KubeVirt em execução na Virtualização OpenShift. Você pode proteger todas as VMs em um ou mais namespaces de uma vez, ou usar o campo `includedVirtualMachines` para selecionar VMs específicas pelo nome. Trident Protect coleta automaticamente todos os recursos dependentes da VM durante o backup e, durante a restauração, você pode recuperar todo o arquivo ou restaurar VMs individualmente.

KubeVirt modos de proteção

Trident Protect oferece duas maneiras de definir quais KubeVirt VMs proteger. Escolha um modo ao definir um aplicativo:

Modo	Campo	Descrição
Baseado em namespace	`includedNamespaces`	Proteja todos os recursos correspondentes em um ou mais namespaces. Use um `labelSelector` na entrada do namespace para controlar quais VMs serão incluídas dinamicamente.
Com escopo de VM	`includedVirtualMachines`	Proteja máquinas virtuais KubeVirt específicas por namespace e nome. Use este modo quando as máquinas virtuais estiverem distribuídas por namespaces ou quando você não quiser proteger todos os recursos de um namespace.

Você deve especificar exatamente um campo (`includedNamespaces` ou `includedVirtualMachines`) no `ApplicationSpec`. Você não pode usar ambos na mesma aplicação.

Ambos os modos suportam filtragem adicional:

- Use `resourceFilter` para incluir ou excluir recursos específicos.
- Use `includedClusterScopedResources` para incluir recursos compatíveis com escopo de cluster, como `StorageClass` objetos.

OBSERVAÇÃO

- Trident Protect pode congelar e descongelar os sistemas de arquivos de máquinas virtuais KubeVirt durante operações de proteção de dados para garantir a consistência. Para detalhes de configuração, consulte "Proteção de dados com máquinas virtuais do KubeVirt".
- Para criar snapshots de todos os discos de uma VM no mesmo ponto no tempo, use snapshots de grupo de volume. Consulte "Use snapshots de grupo de volume com Trident Protect".

Recursos coletados durante backup e snapshot

Ao criar um backup ou snapshot para um aplicativo que inclui KubeVirt VMs, Trident Protect coleta os recursos de VM selecionados e descobre automaticamente seus recursos dependentes.

O comportamento de coleta se aplica em dois casos:

- Aplicações com escopo de VM que usam `includedVirtualMachines`.
- Aplicações baseadas em namespaces que contêm KubeVirt `VirtualMachine` recursos.

Coleção com escopo de VM

Para aplicações com escopo de máquina virtual (VM), Trident Protect inicia com cada `VirtualMachine` listada em `includedVirtualMachines` e descobre os recursos dependentes necessários para restaurar essa VM.

Tipo de recurso	Escopo	Como é descoberto
`VirtualMachine`	Com namespace	Correspondência direta da lista `includedVirtualMachines`.
`VirtualMachineInstance`	Com namespace	Mesmo namespace e nome da VM, se a VM estiver em execução.
`VirtualMachineInstancetype`	Com namespace	Referenciado por `spec.instancetype` quando `kind: VirtualMachineInstancetype`.
`VirtualMachineClusterInstancetype`	Cluster	Referenciado por `spec.instancetype` quando `kind` não está definido ou `kind: VirtualMachineClusterInstancetype`.
`VirtualMachinePreference`	Com namespace	Referenciado por `spec.preference` quando `kind: VirtualMachinePreference`.
`VirtualMachineClusterPreference`	Cluster	Referenciado por `spec.preference` quando `kind` não está definido ou `kind: VirtualMachineClusterPreference`.
`DataVolume`	Com namespace	Referenciado por `spec.template.spec.volumes[].dataVolume.name`.

Tipo de recurso	Escopo	Como é descoberto
`PersistentVolumeClaim`	Com namespace	Descoberto a partir de PVCs de suporte de DataVolume, que usam o mesmo nome que o DataVolume; referências diretas de PVC em `persistentVolumeClaim.claimName`; PVCs de despejo de memória em `memoryDump.claimName`; e PVCs efêmeros em `ephemeral.persistentVolumeClaim.claimName`.
`PersistentVolume`	Cluster	Descoberto a partir do PV de suporte para cada PVC coletado usando `spec.volumeName`.
`Secret`	Com namespace	Descoberto a partir de segredos de volume em `secret.secretName`; segredos de extração de imagem containerDisk; referências a segredos de dados do usuário e de rede CloudInitNoCloud e CloudInitConfigDrive; segredos Sysprep; segredos de extração de imagem de contêiner KernelBoot; e segredos de SSH ou senha AccessCredentials.
`ConfigMap`	Com namespace	Descoberto a partir do volume ConfigMaps em `configMap.name` e Sysprep ConfigMaps.
`ServiceAccount`	Com namespace	Descoberto a partir de `serviceAccount.serviceAccountName`referências de volume.
`NetworkAttachmentDefinition`	Com namespace	Descoberto em redes Multus `spec.template.spec.networks[].multus.networkName` quando o nome da rede não contém `/`. Somente referências NetworkAttachmentDefinition do mesmo namespace são coletadas. Referências entre namespaces diferentes são ignoradas.
`Pod`	Com namespace	Descoberto para `virt-launcher`Pods usando o seletor de rótulos `kubevirt.io=virt-launcher,vm.kubevirt.io/name=<vmName>`.
Classe de armazenamento, volumes Trident e backends Trident	Cluster	Coletados da mesma forma que os aplicativos baseados em namespace, com base nos PVCs descobertos para a VM.

OBSERVAÇÃO	Após a coleta de todos os recursos com escopo de máquina virtual e namespace, Trident Protect aplica o padrão `resourceFilter`, caso esteja configurado. A coleta de recursos com escopo de cluster prossegue normalmente.

Coleção baseada em namespace

Para aplicações baseadas em namespaces, Trident Protect examina os namespaces selecionados e coleta todos os recursos correspondentes. Se algum KubeVirt `VirtualMachine` objeto for encontrado, Trident Protect coleta automaticamente seus recursos dependentes, incluindo tipos de instância, preferências e `virt-launcher` Pods, da mesma forma que faz para aplicações com escopo de máquina virtual.

Isso significa que os recursos específicos da VM são incluídos no seu backup ou snapshot mesmo quando o aplicativo é definido com `includedNamespaces`.

Proteja máquinas virtuais usando definições de aplicativos baseadas em namespace

Ao criar um aplicativo baseado em namespaces, Trident Protect examina os namespaces selecionados e coleta todos os recursos, incluindo quaisquer KubeVirt VMs. Para cada máquina virtual encontrada, Trident Protect também coleta os recursos dependentes necessários para fazer backup e restaurar essa máquina virtual.

Exemplo de CR de aplicação baseada em namespace:

```
apiVersion: protect.trident.netapp.io/v1
kind: Application
metadata:
  name: <application_name>
  namespace: <application_namespace>
spec:
  includedNamespaces:
    - namespace: <vm_namespace>
```

Após criar o aplicativo, use os fluxos de trabalho padrão do Trident Protect para backups e snapshots:

- Para exemplos de snapshots em YAML, veja "Crie um snapshot sob demanda".
- Para exemplos de YAML de backup, consulte "Crie um backup sob demanda".

Comportamento de restauração para aplicativos de máquina virtual baseados em namespace

Como os aplicativos baseados em namespaces coletam recursos dos namespaces selecionados, o backup ou snapshot pode incluir objetos de tempo de execução do KubeVirt e metadados gerenciados pelo KubeVirt. Alguns desses objetos não devem ser restaurados diretamente porque representam o estado de tempo de execução do cluster de origem.

Durante a restauração, Trident Protect aplica transformações compatíveis com KubeVirt para remover os metadados do cluster de origem e ignorar recursos transitórios do KubeVirt. Isso ajuda o KubeVirt a recriar a máquina virtual corretamente no ambiente restaurado.

Recurso	Transformar
VirtualMachine	Remove rótulos e anotações que começam com `kubevirt.io/`.
VirtualMachineInstance	Remove rótulos e anotações que começam com `kubevirt.io/`.
VirtualMachineInstanceMigration	Ignorado e não restaurado.
VirtualMachineSnapshot	Ignorado e não restaurado.
VirtualMachineSnapshotContent	Ignorado e não restaurado.
Pod (`virt-launcher`)	Ignorado quando o pod possui o `kubevirt.io=virt-launcher` rótulo.

OBSERVAÇÃO

- Trident Protect restaura os recursos do KubeVirt em uma ordem definida para evitar problemas de dependência. CDI DataVolumes (`cdi.kubevirt.io` são restaurados primeiro para que os dados de disco da máquina virtual estejam disponíveis antes da criação dos objetos da máquina virtual. Depois disso,

VirtualMachines (`kubevirt.io` são restauradas antes de VirtualMachineInstances.

- Por padrão, o Trident Protect remove o endereço MAC da primeira interface de rede de cada VM restaurada. Para preservá-lo, utilize a `protect.trident.netapp.io/retain-mac-address` anotação. Consulte "Utilize as configurações avançadas de restauração do Trident Protect.".

Proteção dinâmica de VM com seletores de rótulos

Se você deseja proteger máquinas virtuais dinamicamente sem listar o nome de cada máquina virtual, use definições de aplicativos baseadas em namespace com seletores de rótulo.

Apenas o VirtualMachine CR precisa de rótulos. Recursos dependentes, como DataVolumes, PVCs, Segredos e ConfigMaps, são coletados automaticamente.

Passos

1. Rotule as CRs da máquina virtual com um rótulo compartilhado:

```
kubectl label virtualmachine <vm_name 1> app=<label_value> -n
<vm_namespace>
kubectl label virtualmachine <vm_name_2> app=<label_value> -n
<vm_namespace>
```

2. Crie um CR de aplicativo com `includedNamespaces` e um `labelSelector`:

```
apiVersion: protect.trident.netapp.io/v1
kind: Application
metadata:
  name: <application_name>
  namespace: <application_namespace>
spec:
  includedNamespaces:
    - namespace: <vm_namespace>
      labelSelector:
        matchLabels:
          app: <label_value>
```

3. Crie backups e restaurações usando o fluxo de trabalho padrão do Trident Protect.

Detalhes adicionais:

- Você pode adicionar ou remover VMs da proteção a qualquer momento alterando os rótulos:

```
kubectl label virtualmachine <vm_name_to_add> app=<label_value> -n
<vm_namespace>
kubectl label virtualmachine <vm_name_to_remove> app
```

```
=<different_label_value> -n <vm_namespace>
```

- Você também pode usar seletores de rótulo em vários namespaces:

```
spec:
  includedNamespaces:
    - namespace: <vm_namespace_1>
      labelSelector:
        matchLabels:
          <label_key>: <label_value>
    - namespace: <vm_namespace_2>
      labelSelector:
        matchLabels:
          <label_key>: <label_value>
```

Proteja VMs específicas com includedVirtualMachines

Use `includedVirtualMachines` quando você quiser proteger um conjunto nomeado de VirtualMachines em um ou mais namespaces e restaurar seletivamente VMs específicas de um arquivamento.

Tanto a solicitação de alteração (CR) quanto a restauração utilizam o mesmo formato de seletor `includedVirtualMachines`.

Utilize este seletor em diferentes contextos:

- Em um CR de aplicação, ele define quais VMs estão protegidas.
- Em uma solicitação de restauração (CR), define quais máquinas virtuais (VMs) são restauradas a partir do backup ou do arquivamento de Snapshot.

OBSERVAÇÃO	Na restauração de CRs, `includedVirtualMachines` é opcional. Se você não incluir este campo, todo o arquivamento será restaurado. Quando especificado, somente as VMs listadas e seus recursos dependentes serão restaurados.

O YAML abaixo mostra a estrutura do seletor compartilhado:

```
includedVirtualMachines:
  - namespace: <vm_namespace_1>
    names:
      - <vm_name_1>
      - <vm_name_2>
  - namespace: <vm_namespace_2>
    names:
      - <vm_name_3>
```

OBSERVAÇÃO	Várias entradas em `includedVirtualMachines` podem referenciar namespaces diferentes, o que possibilita definições e restaurações de aplicativos com

reconhecimento de máquina virtual em múltiplos namespaces.

Campo	Tipo	Descrição
namespace	cadeia de carateres	Namespace do Kubernetes que contém os objetos da máquina virtual de destino.
nomes	lista de strings	Um ou mais nomes de máquinas virtuais nesse namespace.

Exemplo de aplicação baseada em máquina virtual:

```
apiVersion: protect.trident.netapp.io/v1
kind: Application
metadata:
  name: <application_name>
  namespace: <application_namespace>
spec:
  includedVirtualMachines:
    - namespace: <vm_namespace_1>
      names:
        - <vm_name_1>
        - <vm_name_2>
    - namespace: <vm_namespace_2>
      names:
        - <vm_name_3>
  includedClusterScopedResources:
    - group: storage.k8s.io
      kind: StorageClass
```

Restaurar VMs específicas do KubeVirt

Você pode usar `includedVirtualMachines` em CRs de restauração para restaurar apenas as KubeVirt VMs selecionadas de um backup, Snapshot ou arquivamento de Snapshot replicado.

Quando `includedVirtualMachines` é especificado em um CR de restauração, Trident Protect restaura apenas as VirtualMachines listadas e seus recursos dependentes, como `VirtualMachineInstance`, `DataVolume`, `PersistentVolumeClaim`, `PersistentVolume`, `Secret`, `ConfigMap`, `ServiceAccount` e outros objetos relacionados à máquina virtual.

Se você não especificar `includedVirtualMachines`na CR de restauração, o Trident Protect restaura o arquivamento completo de acordo com a configuração da CR de restauração.

CRs de restauração e comandos CLI compatíveis

A tabela a seguir lista os CRs de restauração compatíveis `includedVirtualMachines` e seus respectivos comandos da CLI.

Restaurar CR	Comando CLI	Descrição
`BackupRestore`	`tridentctl-protect create backuprestore`	Restaura a partir de um backup para mapeamentos de namespace novos ou diferentes. Suporta `namespaceMapping`e `storageClassMapping`.
`BackupInplaceRestore`	`tridentctl-protect create backupinplacerestore`	Restaura a partir de um backup no namespace ou namespaces originais.
`SnapshotRestore`	`tridentctl-protect create snapshotrestore`	Restaura a partir de um snapshot para mapeamentos de namespace novos ou diferentes. Suporta `namespaceMapping` e `storageClassMapping`.
`SnapshotInplaceRestore`	`tridentctl-protect create snapshotinplacerestore`	Restaura a partir de um Snapshot no namespace ou namespaces originais.
`ReplicateSnapshotRestore`	Somente YAML ou use o comando CLI compatível se disponível em seu ambiente	Restaura a partir de um snapshot replicado. Suporta `namespaceMapping`, `storageClassMapping` e restauração no local usando a flag `inPlaceRestore`.

OBSERVAÇÃO	A `tridentctl-protect`CLI não possui uma flag para `includedVirtualMachines`em comandos de restauração. Para restaurar VMs específicas, você deve aplicar um arquivo YAML de restauração CR diretamente usando `kubectl apply`. Você pode escrever o YAML do zero ou usar a CLI com `--dry-run --output yaml`para gerar um arquivo inicial, adicionar o campo `includedVirtualMachines` em `spec`, e então aplicá-lo.

Comportamento de restauração com escopo de VM

O campo `includedVirtualMachines` em um CR de restauração usa o mesmo formato de seletor que o CR do aplicativo:

```
includedVirtualMachines:
  - namespace: <source_vm_namespace>
    names:
      - <vm_name_1>
      - <vm_name_2>
```

Para CRs de restauração, os valores de namespace `includedVirtualMachines` referem-se aos namespaces de origem conforme existem no backup, Snapshot ou arquivamento de Snapshot replicado.

Se você usar `namespaceMapping`, o Trident Protect primeiro seleciona as máquinas virtuais solicitadas do arquivamento de origem e, em seguida, aplica os mapeamentos de namespace durante a restauração. Se você usar `storageClassMapping`, o Trident Protect aplica os mapeamentos de classe de armazenamento durante a fase de transformação da restauração.

Gerar YAML de restauração com a CLI

Você pode usar `tridentctl-protect` para gerar um CR de restauração base e, em seguida, editar o YAML para adicionar `includedVirtualMachines`.

```
tridentctl-protect create <restore_command> <restore_name> \
  <restore_options> \
  --dry-run --output yaml > <restore_name>.yaml
```

Edite o arquivo YAML gerado e adicione o campo `includedVirtualMachines` em `spec`, depois aplique o arquivo:

```
kubectl apply -f <restore_name>.yaml
```

Exemplo: defina um aplicativo baseado em máquina virtual

O seguinte aplicativo protege três máquinas virtuais específicas em dois namespaces.

```
apiVersion: protect.trident.netapp.io/v1
kind: Application
metadata:
  name: production-vms
  namespace: prod
spec:
  includedVirtualMachines:
    - namespace: prod
      names:
        - app-server
        - database-server
    - namespace: monitoring
      names:
        - prometheus-vm
  includedClusterScopedResources:
    - group: storage.k8s.io
      kind: StorageClass
```

Exemplo: criar um backup

A criação de um backup para um aplicativo baseado em máquina virtual utiliza o backup CR padrão. Não são necessários campos específicos de máquina virtual no backup CR.

```
apiVersion: protect.trident.netapp.io/v1
kind: Backup
metadata:
  name: daily-backup
```

```
  namespace: prod
spec:
  applicationRef: production-vms
  appVaultRef: s3-vault
```

Exemplo: restaurar uma única máquina virtual a partir de um backup

O exemplo a seguir restaura apenas `database-server` do arquivamento de backup. As outras VMs no arquivamento, como `app-server` e `prometheus-vm`, são ignoradas.

```
apiVersion: protect.trident.netapp.io/v1
kind: BackupRestore
metadata:
  name: restore-db-only
  namespace: prod-vms-restore
spec:
  appVaultRef: s3-vault
  appArchivePath: backups/production-vms/2026-03-10T00-00-00Z
  namespaceMapping:
    - source: prod
      destination: prod-dr
  includedVirtualMachines:
    - namespace: prod
      names:
        - database-server
```

```
kubectl apply -f restore-db-only.yaml
```

Somente `database-server` e seus recursos dependentes, como DataVolumes, PVCs, PVs, Secrets, ConfigMaps e recursos relacionados do KubeVirt, são restaurados.

Exemplo: restaurar uma única máquina virtual no local a partir de um backup

O exemplo a seguir restaura apenas `database-server`no namespace original a partir do arquivamento de backup.

```
apiVersion: protect.trident.netapp.io/v1
kind: BackupInplaceRestore
metadata:
  name: ipr-db-only
  namespace: prod
spec:
  appVaultRef: s3-vault
  appArchivePath: backups/production-vms/2026-03-10T00-00-00Z
```

```
  includedVirtualMachines:
    - namespace: prod
      names:
        - database-server
```

```
kubectl apply -f ipr-db-only.yaml
```

Exemplo: restaurar uma única máquina virtual a partir de um Snapshot

O exemplo a seguir restaura apenas `ubuntu-vm-blue-a627be38` de um arquivamento de Snapshot.

```
apiVersion: protect.trident.netapp.io/v1
kind: SnapshotRestore
metadata:
  name: blue-vm-snap-restore
  namespace: blue
spec:
  appVaultRef: my-appvault
  appArchivePath: snapshots/protectctl-blue-vm/snap-1
  includedVirtualMachines:
    - namespace: blue
      names:
        - ubuntu-vm-blue-a627be38
```

```
kubectl apply -f blue-vm-snap-restore.yaml
```

Combine a restauração da máquina virtual com outros filtros

Você pode usar `includedVirtualMachines` junto com `resourceFilter`, `namespaceMapping` e `storageClassMapping` no mesmo CR de restauração para ajustar com precisão o que será restaurado.

Ao adicionar um `resourceFilter`, Trident Protect primeiro seleciona as VMs listadas em `includedVirtualMachines` e seus recursos dependentes. Em seguida, aplica o filtro de recursos para restringir ainda mais o que está incluído ou excluído desses resultados.

```
spec:
  includedVirtualMachines:
    - namespace: <vm_namespace>
      names:
        - <vm_name>
  resourceFilter:
    resourceSelectionCriteria: Include
    resourceMatchers:
```

```
    - kinds: ["PersistentVolumeClaim"]
      names: ["<pvc_name>"]
```

Para `BackupRestore`, `SnapshotRestore` e `ReplicateSnapshotRestore`, `namespaceMapping` e `storageClassMapping` funcionam junto com `includedVirtualMachines`. As referências de namespace em `includedVirtualMachines` correspondem aos namespaces de origem conforme aparecem no arquivamento. Os mapeamentos de namespace e classe de armazenamento são aplicados durante a fase de transformação de restauração após a conclusão da filtragem.

Para obter detalhes sobre o comportamento de restauração, consulte "Restaure aplicativos usando o Trident Protect.".

Crie um aplicativo com escopo de máquina virtual usando a CLI

Use a `--virtual-machines` flag para restringir o escopo de um aplicativo a KubeVirt VMs específicas em vez de namespaces inteiros. A `--virtual-machines` flag é mutuamente exclusiva com `--namespaces`.

Passos

1. Crie a aplicação executando um dos seguintes comandos, substituindo os valores entre colchetes pelas informações do seu ambiente. O formato para o `--virtual-machines` parâmetro é `<namespace>(<vm_name>)`. Para várias VMs, use nomes separados por vírgulas dentro dos parênteses:
   - Uma única máquina virtual em um namespace:

     ```
     tridentctl-protect create application <application_name> \
       -n <application_namespace> \
       --virtual-machines "<vm_namespace>(<vm_name>)"
     ```

   - Múltiplas máquinas virtuais em diferentes namespaces:

     ```
     tridentctl-protect create application <application_name> \
       -n <application_namespace> \
       --virtual-machines "<vm_namespace_1>(<vm_name_1>,<vm_name_2>
     ),<vm_namespace_2>(<vm_name_3>)"
     ```

Informações relacionadas

- "Definir uma aplicação para gestão"
- "Proteger aplicações"
- "Restaure aplicativos usando o Trident Protect."
- "Use as configurações avançadas de restauração"
- "Replique aplicações usando NetApp SnapMirror e Trident Protect."
- "Instale o plugin Trident Protect CLI"
- "Use objetos AppVault para gerenciar buckets"

Gerenciar os ganchos de execução do Trident Protect

Um gancho de execução é uma ação personalizada que você pode configurar para ser executada em conjunto com uma operação de proteção de dados de um aplicativo gerenciado. Por exemplo, se você tiver um aplicativo de banco de dados, poderá usar um gancho de execução para pausar todas as transações de banco de dados antes de um snapshot e retomar as transações após a conclusão do snapshot. Isso garante snapshots consistentes com aplicativos.

Tipos de ganchos de execução

O Trident Protect suporta os seguintes tipos de ganchos de execução, com base em quando podem ser executados:

- Pré-instantâneo
- Pós-snapshot
- Pré-backup
- Pós-backup
- Pós-restauração
- Pós-failover

Ordem de execução

Quando uma operação de proteção de dados é executada, os eventos de gancho de execução ocorrem na seguinte ordem:

1. Todos os ganchos de execução personalizados de pré-operação aplicáveis são executados nos contentores apropriados. Você pode criar e executar quantos ganchos de pré-operação personalizados você precisar, mas a ordem de execução desses ganchos antes da operação não é garantida nem configurável.
2. Congelamentos do sistema de arquivos ocorrem, se aplicável. "Saiba mais sobre como configurar o congelamento do sistema de arquivos com o Trident Protect.".
3. A operação de proteção de dados é realizada.
4. Os sistemas de arquivos congelados não estão congelados, se aplicável.
5. Todos os ganchos de execução pós-operação personalizados aplicáveis são executados nos contentores apropriados. Você pode criar e executar quantos ganchos de pós-operação personalizados você precisar, mas a ordem de execução desses ganchos após a operação não é garantida nem configurável.

Se você criar vários ganchos de execução do mesmo tipo (por exemplo, pré-snapshot), a ordem de execução desses ganchos não será garantida. No entanto, a ordem de execução de ganchos de diferentes tipos é garantida. Por exemplo, a seguinte é a ordem de execução de uma configuração que tem todos os diferentes tipos de ganchos:

1. Ganchos pré-instantâneos executados
2. Ganchos pós-snapshot executados
3. Ganchos pré-backup executados
4. Ganchos pós-backup executados

OBSERVAÇÃO	O exemplo de pedido anterior só se aplica quando você executa um backup que não usa um snapshot existente.
OBSERVAÇÃO	Você deve sempre testar seus scripts de gancho de execução antes de habilitá-los em um ambiente de produção. Você pode usar o comando 'kubectl exec' para testar convenientemente os scripts. Depois de habilitar os ganchos de execução em um ambiente de produção, teste os snapshots e backups resultantes para garantir que eles sejam consistentes. Você pode fazer isso clonando o aplicativo para um namespace temporário, restaurando o snapshot ou o backup e testando o aplicativo.
OBSERVAÇÃO	Se um gancho de execução pré-snapshot adicionar, alterar ou remover recursos do Kubernetes, essas alterações serão incluídas no snapshot ou backup e em qualquer operação de restauração subsequente.

Notas importantes sobre ganchos de execução personalizados

Considere o seguinte ao Planejar ganchos de execução para seus aplicativos.

- Um gancho de execução deve usar um script para executar ações. Muitos ganchos de execução podem referenciar o mesmo script.
- O Trident Protect exige que os scripts usados pelos ganchos de execução sejam escritos no formato de scripts shell executáveis.
- O tamanho do script está limitado a 96kbMB.
- O Trident Protect utiliza as configurações de ganchos de execução e quaisquer critérios correspondentes para determinar quais ganchos são aplicáveis a uma operação de snapshot, backup ou restauração.

OBSERVAÇÃO	Como os ganchos de execução geralmente reduzem ou desativam completamente a funcionalidade do aplicativo em que estão sendo executados, você deve sempre tentar minimizar o tempo que seus ganchos de execução personalizados demoram para serem executados. Se você iniciar uma operação de backup ou snapshot com ganchos de execução associados, mas depois cancelá-la, os ganchos ainda poderão ser executados se a operação de backup ou snapshot já tiver começado. Isso significa que a lógica usada em um gancho de execução pós-backup não pode assumir que o backup foi concluído.

Filtros de gancho de execução

Quando você adiciona ou edita um gancho de execução para um aplicativo, você pode adicionar filtros ao gancho de execução para gerenciar quais contentores o gancho corresponderá. Os filtros são úteis para aplicativos que usam a mesma imagem de contentor em todos os contentores, mas podem usar cada imagem para um propósito diferente (como o Elasticsearch). Os filtros permitem criar cenários onde os ganchos de execução são executados em alguns, mas não necessariamente em todos os contentores idênticos. Se você criar vários filtros para um único gancho de execução, eles serão combinados com um operador LÓGICO E. Você pode ter até 10 filtros ativos por gancho de execução.

Cada filtro que você adiciona a um gancho de execução usa uma expressão regular para corresponder aos contêineres no seu cluster. Quando um gancho corresponde a um contêiner, o gancho executará seu script associado naquele contêiner. Expressões regulares para filtros usam a sintaxe Regular Expression 2 (RE2), que não oferece suporte à criação de um filtro que exclua contêineres da lista de correspondências. Para obter informações sobre a sintaxe que o Trident Protect suporta para expressões regulares em filtros de

gancho de execução, consulte "Suporte à sintaxe da expressão regular 2 (RE2)" .

OBSERVAÇÃO	Se você adicionar um filtro de namespace a um gancho de execução que é executado após uma operação de restauração ou clone e a origem e destino de restauração ou clone estiverem em namespaces diferentes, o filtro de namespace será aplicado somente ao namespace de destino.

Exemplos de gancho de execução

Visite o "Projeto NetApp Verda GitHub" para baixar ganchos de execução reais para aplicativos populares, como Apache Cassandra e Elasticsearch. Você também pode ver exemplos e obter ideias para estruturar seus próprios ganchos de execução personalizados.

OBSERVAÇÃO	Os scripts no repositório Verda são fornecidos "como estão" e não são oficialmente suportados pela NetApp. O suporte da NetApp abrange apenas a estrutura de execução de hooks dentro do Trident Protect.

Crie um gancho de execução

Você pode criar um gancho de execução personalizado para um aplicativo usando o Trident Protect. Você precisa ter permissões de Proprietário, Administrador ou Membro para criar ganchos de execução.

Use um CR

Passos

1. Crie o arquivo de recurso personalizado (CR) e nomeie-o `trident-protect-hook.yaml`.
2. Configure os seguintes atributos para corresponder ao seu ambiente Trident Protect e à configuração do cluster:
   - **metadata.name**: (*required*) o nome deste recurso personalizado; escolha um nome único e sensível para o seu ambiente.
   - **Spec.applicationRef**: (*required*) o nome do Kubernetes do aplicativo para o qual executar o gancho de execução.
   - **Spec.stage**: (*required*) Uma cadeia de carateres indicando qual estágio durante a ação o gancho de execução deve ser executado. Valores possíveis:
     - Pre
     - Post
   - **Spec.action**: (*required*) Uma cadeia de carateres indicando qual ação o gancho de execução tomará, supondo que quaisquer filtros de gancho de execução especificados sejam correspondentes. Valores possíveis:
     - Snapshot
     - Backup
     - Restaurar
     - Failover
   - **Spec.enabled**: (*Optional*) indica se esse gancho de execução está ativado ou desativado. Se não for especificado, o valor padrão é verdadeiro.
   - **Spec.hookSource**: (*required*) Uma string contendo o script de gancho codificado em base64.

- **Spec.timeout**: (*Optional*) Um número que define quanto tempo em minutos o gancho de execução pode ser executado. O valor mínimo é de 1 minuto e o valor padrão é de 25 minutos, se não for especificado.
- **Spec.arguments**: (*Optional*) Uma lista YAML de argumentos que você pode especificar para o gancho de execução.
- **Spec.matchingCriteria**: (*Optional*) uma lista opcional de pares de valores de chave de critérios, cada par compondo um filtro de gancho de execução. Você pode adicionar até 10 filtros por gancho de execução.
- **Spec.matchingCriteria.type**: (*Optional*) Uma string que identifica o tipo de filtro do gancho de execução. Valores possíveis:
  - ContainerImage
  - Nome do ConteinerName
  - PodName
  - PodLabel
  - NamespaceName
- **Spec.matchingCriteria.value**: (*Optional*) Uma string ou expressão regular identificando o valor do filtro do gancho de execução.

  Exemplo YAML:

```
apiVersion: protect.trident.netapp.io/v1
kind: ExecHook
metadata:
  name: example-hook-cr
  namespace: my-app-namespace
  annotations:
    astra.netapp.io/astra-control-hook-source-id:
/account/test/hookSource/id
spec:
  applicationRef: my-app-name
  stage: Pre
  action: Snapshot
  enabled: true
  hookSource: IyEvYmluL2Jhc2gKZWNobyAiZXhhbXBsZSBzY3JpcHQiCg==
  timeout: 10
  arguments:
    - FirstExampleArg
    - SecondExampleArg
  matchingCriteria:
    - type: containerName
      value: mysql
    - type: containerImage
      value: bitnami/mysql
    - type: podName
      value: mysql
```

```
    - type: namespaceName
      value: mysql-a
    - type: podLabel
      value: app.kubernetes.io/component=primary
    - type: podLabel
      value: helm.sh/chart=mysql-10.1.0
    - type: podLabel
      value: deployment-type=production
```

3. Depois de preencher o ficheiro CR com os valores corretos, aplique o CR:

```
kubectl apply -f trident-protect-hook.yaml
```

Use a CLI

Passos

1. Crie o gancho de execução, substituindo valores entre parênteses por informações do seu ambiente. Por exemplo:

```
tridentctl-protect create exechook <my_exec_hook_name> --action
<action_type> --app <app_to_use_hook> --stage <pre_or_post_stage>
--source-file <script-file> -n <application_namespace>
```

Execute manualmente um gancho de execução

Você pode executar manualmente um gancho de execução para fins de teste ou se precisar executar novamente o gancho manualmente após uma falha. Você precisa ter permissões de proprietário, administrador ou membro para executar manualmente os ganchos de execução.

Executar manualmente um gancho de execução consiste em duas etapas básicas:

1. Crie um backup de recursos, que coleta recursos e cria um backup deles, determinando onde o gancho será executado
2. Execute o gancho de execução contra o backup

▼ Passo 1: Crie um backup de recursos

Use um CR

Passos

1. Crie o arquivo de recurso personalizado (CR) e nomeie-o `trident-protect-resource-backup.yaml`.
2. Configure os seguintes atributos para corresponder ao seu ambiente Trident Protect e à configuração do cluster:
   - **metadata.name**: (*required*) o nome deste recurso personalizado; escolha um nome único e sensível para o seu ambiente.

- **Spec.applicationRef**: (*required*) o nome do Kubernetes do aplicativo para o qual criar o backup de recursos.
- **Spec.appVaultRef**: (*required*) o nome do AppVault onde o conteúdo de backup é armazenado.
- **Spec.appArchivePath**: O caminho dentro do AppVault onde o conteúdo do backup é armazenado. Você pode usar o seguinte comando para encontrar este caminho:

```
kubectl get backups <BACKUP_NAME> -n my-app-namespace -o
jsonpath='{.status.appArchivePath}'
```

Exemplo YAML:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: ResourceBackup
metadata:
  name: example-resource-backup
spec:
  applicationRef: my-app-name
  appVaultRef: my-appvault-name
  appArchivePath: example-resource-backup
```

3. Depois de preencher o ficheiro CR com os valores corretos, aplique o CR:

```
kubectl apply -f trident-protect-resource-backup.yaml
```

Use a CLI

Passos

1. Crie o backup, substituindo valores entre parênteses por informações do seu ambiente. Por exemplo:

```
tridentctl protect create resourcebackup <my_backup_name> --app
<my_app_name> --appvault <my_appvault_name> -n <my_app_namespace>
--app-archive-path <app_archive_path>
```

2. Ver o estado da cópia de segurança. Você pode usar este comando de exemplo repetidamente até que a operação esteja concluída:

```
tridentctl protect get resourcebackup -n <my_app_namespace>
<my_backup_name>
```

3. Verifique se o backup foi bem-sucedido:

```
kubectl describe resourcebackup <my_backup_name>
```

▼ **Passo 2: Execute o gancho de execução**

Use um CR

Passos

1. Crie o arquivo de recurso personalizado (CR) e nomeie-o `trident-protect-hook-run.yaml`.
2. Configure os seguintes atributos para corresponder ao seu ambiente Trident Protect e à configuração do cluster:
   - **metadata.name**: (*required*) o nome deste recurso personalizado; escolha um nome único e sensível para o seu ambiente.
   - **Spec.applicationRef**: (*required*) Certifique-se de que este valor corresponde ao nome da aplicação do ResourceBackup CR criado na etapa 1.
   - **Spec.appVaultRef**: (*required*) Certifique-se de que este valor corresponde ao appVaultRef do ResourceBackup CR criado na etapa 1.
   - **Spec.appArchivePath**: Certifique-se de que este valor corresponda ao appArchivePath do ResourceBackup CR criado na etapa 1.

     ```
     kubectl get backups <BACKUP_NAME> -n my-app-namespace -o
     jsonpath='{.status.appArchivePath}'
     ```

   - **Spec.action**: (*required*) Uma cadeia de carateres indicando qual ação o gancho de execução tomará, supondo que quaisquer filtros de gancho de execução especificados sejam correspondentes. Valores possíveis:
     - Snapshot
     - Backup
     - Restaurar
     - Failover
   - **Spec.stage**: (*required*) Uma cadeia de carateres indicando qual estágio durante a ação o gancho de execução deve ser executado. Esta corrida de gancho não vai correr ganchos em qualquer outro estágio. Valores possíveis:
     - Pre
     - Post

     Exemplo YAML:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: ExecHooksRun
metadata:
  name: example-hook-run
```

```
spec:
  applicationRef: my-app-name
  appVaultRef: my-appvault-name
  appArchivePath: example-resource-backup
  stage: Post
  action: Failover
```

3. Depois de preencher o ficheiro CR com os valores corretos, aplique o CR:

```
kubectl apply -f trident-protect-hook-run.yaml
```

Use a CLI

Passos

1. Crie a solicitação de execução manual do hook run:

```
tridentctl protect create exechooksrun <my_exec_hook_run_name> -n
<my_app_namespace> --action snapshot --stage <pre_or_post> --app
<my_app_name> --appvault <my_appvault_name> --path
<my_backup_name>
```

2. Verifique o status da execução do hook run. Você pode executar este comando repetidamente até que a operação esteja concluída:

```
tridentctl protect get exechooksrun -n <my_app_namespace>
<my_exec_hook_run_name>
```

3. Descreva o objeto exechooksrun para ver os detalhes e status finais:

```
kubectl -n <my_app_namespace> describe exechooksrun
<my_exec_hook_run_name>
```

Desinstale o Trident Protect.

Você pode precisar remover componentes do Trident Protect se estiver atualizando de uma versão de avaliação para a versão completa do produto.

Para remover o Trident Protect, siga os passos abaixo.

Passos

1. Remova os arquivos CR do Trident Protect:

OBSERVAÇÃO Esta etapa não é necessária para a versão 25.06 e posteriores.

```
helm uninstall -n trident-protect trident-protect-crds
```

2. Remover Trident Protect:

```
helm uninstall -n trident-protect trident-protect
```

3. Remova o namespace Trident Protect:

```
kubectl delete ns trident-protect
```

Blogs Trident e Trident Protect

Você pode encontrar ótimos blogs sobre NetApp Trident e Trident Protect aqui:

Trident blogs

- 16 de outubro de 2025:"Soluções avançadas de armazenamento para Kubernetes"
- 19 de agosto de 2025:"Aprimorando a consistência de dados: instantâneos de grupos de volume na virtualização OpenShift com Trident"
- 09 de maio de 2025: "Configuração automática do backend Trident para FSx para ONTAP com o complemento Amazon EKS"
- 15 de abril de 2025: "NetApp Trident com Google Cloud NetApp Volumes para protocolo SMB"
- 14 de abril de 2025: "Aproveitando o Protocolo Fibre Channel com o Trident 25.02 para Armazenamento Persistente no Kubernetes"
- 14 de abril de 2025: "Desbloqueando o poder dos sistemas NetApp ASA r2 para armazenamento em bloco do Kubernetes"
- 31 de março de 2025: "Simplificando a instalação do Trident no Red Hat OpenShift com o novo operador certificado"
- 27 de março de 2025: "Provisionamento do Trident para PMEs com o Google Cloud NetApp Volumes"
- 05 de março de 2025: "Desbloquear integração contínua de armazenamento iSCSI: Um guia para FSxN em clusters ROSA para AWS"
- 27 de fevereiro de 2025: "Implantando identidade na nuvem com Trident, GKE e Google Cloud NetApp Volumes"
- 12 de dezembro de 2024: "Apresentação do suporte ao Fibre Channel no Trident"
- 11 de novembro de 2024:"NetApp Trident com Google Cloud NetApp Volumes"
- 29 de outubro de 2024: "Amazon FSx for NetApp ONTAP com Red Hat OpenShift Service na AWS (ROSA) usando Trident"
- 29 de outubro de 2024: "Migração ao vivo de VMs com OpenShift Virtualization no ROSA e Amazon FSx for NetApp ONTAP"
- 08 de julho de 2024: "Usando NVMe/TCP para consumir armazenamento ONTAP para seus aplicativos modernos em contêineres no Amazon EKS"
- 01 de julho de 2024: "Armazenamento Kubernetes integrado com Google Cloud NetApp Volumes Flex e Astra Trident"
- 11 de junho de 2024: "ONTAP como armazenamento de backend para o registro de imagem integrado no OpenShift"

Blogs da Trident Protect

- 16 de maio de 2025:"Automatizando o failover do registro para recuperação de desastres com os hooks pós-restauração do Trident Protect."
- 16 de maio de 2025: "Recuperação de desastres de virtualização OpenShift com NetApp Trident Protect"
- 13 de maio de 2025:"Migração de classe de armazenamento com backup e restauração do Trident Protect."

- 09 de maio de 2025:"Redimensione aplicações Kubernetes com os hooks de pós-restauração do Trident Protect."
- 03 de abril de 2025: "Trident Protect Power Up: Replicação do Kubernetes para Proteção e Recuperação de Desastres"
- 13 de Mar de 2025: "Operações de backup e restauração consistentes com falhas para VMs de virtualização OpenShift"
- 11 de Mar de 2025: "Estendendo padrões GitOps para proteção de dados de aplicativos com o NetApp Trident"
- 03 de Mar de 2025: "Trident 25,02: Elevando a experiência do Red Hat OpenShift com novos recursos emocionantes"
- 15 de janeiro de 2025:"Apresentamos o Trident Protect, um sistema de controle de acesso baseado em funções."
- 11 de novembro de 2024:"Gerenciamento de dados orientado por Kubernetes: a nova era com o Trident Protect."

Conhecimento e apoio

Perguntas frequentes

Encontre respostas para as perguntas mais frequentes sobre instalação, configuração, atualização e solução de problemas do Trident.

Questões gerais

Com que frequência o Trident é lançado?

Começando com o lançamento de 24,02, o Trident é lançado a cada quatro meses: Fevereiro, Junho e Outubro.

O Trident é compatível com todos os recursos lançados em uma versão específica do Kubernetes?

O Trident geralmente não oferece suporte a recursos alfa no Kubernetes. O Trident pode oferecer suporte a recursos beta nas duas versões do Trident que seguem a versão beta do Kubernetes.

O Trident tem alguma dependência de outros produtos NetApp para seu funcionamento?

O Trident não tem dependências em outros produtos de software NetApp e funciona como um aplicativo autônomo. No entanto, você deve ter um dispositivo de storage de back-end do NetApp.

Como posso obter detalhes completos de configuração do Trident?

Use o `tridentctl get` comando para obter mais informações sobre sua configuração do Trident.

Posso obter métricas sobre como o storage é provisionado pelo Trident?

Sim. Endpoints Prometheus que podem ser usados para coletar informações sobre a operação do Trident, como o número de backends gerenciados, o número de volumes provisionados, bytes consumidos e assim por diante. Você também pode usar "Cloud Insights" para monitoramento e análise.

A experiência do usuário muda ao usar o Trident como um supervisor CSI?

Não há alterações no que diz respeito à experiência do usuário e às funcionalidades. O nome do provisionador usado é `csi.trident.netapp.io`. Este método de instalação do Trident é recomendado se você quiser usar todos os novos recursos fornecidos pelas versões atuais e futuras.

Instalar e usar o Trident em um cluster do Kubernetes

O Trident suporta uma instalação offline a partir de um registo privado?

Sim, Trident pode ser instalado offline. Consulte "Saiba mais sobre a instalação do Trident".

Posso instalar o Trident be remotamente?

Sim. O Trident 18,10 e posterior suportam a capacidade de instalação remota de qualquer máquina que tenha `kubectl` acesso ao cluster. Depois `kubectl` que o acesso for verificado (por exemplo, inicie um `kubectl get nodes` comando da máquina remota para verificar), siga as instruções de instalação.

Posso configurar a alta disponibilidade com o Trident?

O Trident é instalado como uma implantação do Kubernetes (ReplicaSet) com uma instância, e por isso tem o HA incorporado. Você não deve aumentar o número de réplicas na implantação. Se o nó em que o Trident está instalado for perdido ou o pod estiver inacessível, o Kubernetes reimplanta automaticamente o pod em um nó íntegro no cluster. O Trident é apenas no plano de controle, portanto os pods atualmente montados não são afetados se o Trident for reimplantado.

O Trident precisa de acesso ao namespace kube-System?

O Trident lê o servidor de API do Kubernetes para determinar quando os aplicativos solicitam novos PVCs, de modo que a TI precisa de acesso ao sistema kube.

Quais são as funções e Privileges usadas pelo Trident?

O instalador do Trident cria um Kubernetes ClusterRole, que tem acesso específico aos PersistentVolume, PersistentVolumeClaim, StorageClass e recursos Secret do cluster Kubernetes. Consulte "Personalize a instalação do tridentctl".

Posso gerar localmente os arquivos de manifesto exatos que o Trident usa para instalação?

Você pode gerar e modificar localmente os arquivos de manifesto exatos que o Trident usa para instalação, se necessário. Consulte "Personalize a instalação do tridentctl".

Posso compartilhar o mesmo SVM de back-end do ONTAP em duas instâncias Trident separadas para dois clusters Kubernetes separados?

Embora não seja aconselhado, você pode usar o mesmo SVM de back-end para duas instâncias do Trident. Especifique um nome de volume exclusivo para cada instância durante a instalação e/ou especifique um parâmetro exclusivo `StoragePrefix` no `setup/backend.json` arquivo. Isso serve para garantir que o mesmo FlexVol volume não seja usado para ambas as instâncias.

É possível instalar o Trident no ContainerLinux (antigo CoreOS)?

O Trident é simplesmente um pod do Kubernetes e pode ser instalado onde quer que o Kubernetes esteja em execução.

Posso usar o Trident com o NetApp Cloud Volumes ONTAP?

Sim, o Trident é compatível com AWS, Google Cloud e Azure.

Solução de problemas e suporte

O NetApp oferece suporte ao Trident?

Embora o Trident seja de código aberto e fornecido gratuitamente, o NetApp o suporta totalmente desde que o back-end do NetApp seja suportado.

Como faço para levantar um caso de suporte?

Para levantar um caso de suporte, execute um dos seguintes procedimentos:

1. Entre em Contato com seu gerente de conta de suporte e obtenha ajuda para levantar um ticket.
2. Envie um caso de suporte entrando em Contato "Suporte à NetApp"com .

Como gerar um pacote de log de suporte?

Você pode criar um pacote de suporte executando `tridentctl logs -a`o . Além dos logs capturados no pacote, capture o log do kubelet para diagnosticar os problemas de montagem no lado do Kubernetes. As instruções para obter o log do kubelet variam de acordo com a forma como o Kubernetes é instalado.

O que devo fazer se for necessário enviar uma solicitação para um novo recurso?

Crie um problema "Trident GitHub" e mencione **RFE** no assunto e na descrição do problema.

Onde posso levantar um defeito?

Crie um problema no "Trident GitHub". Certifique-se de incluir todas as informações e logs necessários relativos ao problema.

O que acontece se eu tiver uma pergunta rápida sobre o Trident que eu preciso de esclarecimentos? Existe uma comunidade ou um fórum?

Se você tiver dúvidas, problemas ou solicitações, entre em Contato conosco através do nosso Trident "Canal discord"ou GitHub.

A senha do meu sistema de armazenamento mudou e o Trident não funciona mais. Como faço para recuperar?

Atualize a senha do backend com `tridentctl update backend myBackend -f </path/to_new_backend.json> -n trident`o . Substitua `myBackend` no exemplo pelo nome do backend e `/path/to_new_backend.json` pelo caminho para o arquivo correto `backend.json`.

O Trident não consegue encontrar meu nó Kubernetes. Como faço para corrigir isso?

Há dois cenários prováveis pelos quais o Trident não consegue encontrar um nó do Kubernetes. Pode ser devido a um problema de rede no Kubernetes ou a um problema de DNS. O daemonset do nó do Trident que é executado em cada nó do Kubernetes deve ser capaz de se comunicar com o controlador Trident para Registrar o nó no Trident. Se as alterações de rede ocorreram após a instalação do Trident, você encontrará esse problema apenas com novos nós do Kubernetes adicionados ao cluster.

Se o pod Trident for destruído, eu perderei os dados?

Os dados não serão perdidos se o pod Trident for destruído. Os metadados do Trident são armazenados em objetos CRD. Todos os PVS que foram provisionados pelo Trident funcionarão normalmente.

Atualize o Trident

Posso atualizar de uma versão mais antiga diretamente para uma versão mais recente (ignorando algumas versões)?

O NetApp suporta a atualização do Trident de uma versão principal para a próxima versão principal imediata. Você pode atualizar da versão 18.xx para 19.xx, 19.xx para 20.xx, e assim por diante. Você deve testar a atualização em um laboratório antes da implantação da produção.

É possível fazer o downgrade do Trident para uma versão anterior?

Se você precisar de uma correção para bugs observados após uma atualização, problemas de dependência ou uma atualização mal sucedida ou incompleta, você deve "Desinstale o Trident"reinstalar a versão anterior

usando as instruções específicas para essa versão. Esta é a única maneira recomendada de fazer o downgrade para uma versão anterior.

Gerenciar backends e volumes

Preciso definir o Gerenciamento e DataLIFs em um arquivo de definição de back-end do ONTAP?

O LIF de gestão é obrigatório. DataLIF varia:

- ONTAP SAN: Não especifique para iSCSI. O Trident usa "Mapa de LUN seletivo da ONTAP" para descobrir os LIFs iSCI necessários para estabelecer uma sessão de vários caminhos. Um aviso é gerado se `dataLIF` for definido explicitamente. "Exemplos e opções de configuração de SAN ONTAP"Consulte para obter detalhes.
- ONTAP nas: A NetApp recomenda especificar `dataLIF`. Se não for fornecido, o Trident buscará os dados LIFs do SVM. Você pode especificar um nome de domínio totalmente qualificado (FQDN) a ser usado para as operações de montagem NFS, permitindo que você crie um DNS round-robin para balanceamento de carga entre vários dataLIFs. "Exemplos e opções de configuração do ONTAP nas"Consulte para obter detalhes

O Trident pode configurar o CHAP para backends ONTAP?

Sim. Trident suporta CHAP bidirecional para backends ONTAP. Isso requer configuração `useCHAP=true` em sua configuração de back-end.

Como faço para gerenciar políticas de exportação com o Trident?

O Trident pode criar e gerenciar dinamicamente políticas de exportação a partir da versão 20,04 em diante. Isso permite que o administrador de storage forneça um ou mais blocos CIDR em sua configuração de back-end e que o Trident adicione IPs de nós que se enquadram nesses intervalos a uma política de exportação criada por ele. Desta forma, o Trident gerencia automaticamente a adição e exclusão de regras para nós com IPs dentro dos CIDR fornecidos.

Os endereços IPv6 podem ser usados para o gerenciamento e DataLIFs?

O Trident suporta a definição de endereços IPv6 para:

- `managementLIF` E `dataLIF` para backends ONTAP nas.
- `managementLIF` Para backends ONTAP SAN. Não é possível especificar `dataLIF` em um back-end de SAN ONTAP.

O Trident deve ser instalado usando o `--use-ipv6` sinalizador (`tridentctl`para instalação), `IPv6` (para o operador Trident) ou `tridentTPv6` (para instalação Helm) para que ele funcione acima de IPv6.

É possível atualizar o LIF de gerenciamento no back-end?

Sim, é possível atualizar o backend Management LIF usando o `tridentctl update backend` comando.

É possível atualizar o DataLIF no backend?

Você pode atualizar o DataLIF em `ontap-nas` e `ontap-nas-economy` somente.

Posso criar vários backends no Trident para Kubernetes?

O Trident pode suportar muitos backends simultaneamente, seja com o mesmo driver ou drivers diferentes.

Como o Trident armazena credenciais de back-end?

O Trident armazena as credenciais de back-end como segredos do Kubernetes.

Como o Trident seleciona um back-end específico?

Se os atributos de back-end não puderem ser usados para selecionar automaticamente os pools corretos para uma classe, os `storagePools` parâmetros e `additionalStoragePools` serão usados para selecionar um conjunto específico de pools.

Como posso garantir que o Trident não provisionará de um back-end específico?

O `excludeStoragePools` parâmetro é usado para filtrar o conjunto de pools que o Trident usa para provisionar e removerá todos os pools correspondentes.

Se houver vários backends do mesmo tipo, como o Trident seleciona qual backend usar?

Se houver vários backends configurados do mesmo tipo, o Trident selecionará o back-end apropriado com base nos parâmetros presentes no `StorageClass` e `PersistentVolumeClaim`no . Por exemplo, se houver vários backends de driver do ONTAP-nas, o Trident tentará corresponder parâmetros `StorageClass` no e `PersistentVolumeClaim` combinou e corresponder a um back-end que possa fornecer os requisitos listados em `StorageClass` e `PersistentVolumeClaim`. Se houver vários backends que correspondam à solicitação, o Trident seleciona um deles aleatoriamente.

O Trident suporta CHAP bidirecional com Element/SolidFire?

Sim.

Como o Trident implementa Qtrees em um volume ONTAP? Quantos Qtrees podem ser implantados em um único volume?

`ontap-nas-economy`O driver cria até 200 Qtrees no mesmo FlexVol volume (configurável entre 50 e 300), 100.000 Qtrees por nó de cluster e 2,4M por cluster. Quando você insere um novo `PersistentVolumeClaim` que é atendido pelo driver de economia, o driver procura ver se já existe um FlexVol volume que pode atender o novo Qtree. Se o FlexVol volume não existir que possa servir o Qtree, um novo FlexVol volume será criado.

Como posso definir permissões Unix para volumes provisionados no ONTAP nas?

Você pode definir permissões Unix no volume provisionado pelo Trident definindo um parâmetro no arquivo de definição de back-end.

Como posso configurar um conjunto explícito de opções de montagem ONTAP NFS enquanto provisiono um volume?

Por padrão, o Trident não define as opções de montagem como nenhum valor com o Kubernetes. Para especificar as opções de montagem na classe de armazenamento do Kubernetes, siga o exemplo fornecido

"aqui".

Como faço para definir os volumes provisionados para uma política de exportação específica?

Para permitir que os hosts apropriados acessem um volume, use o `exportPolicy` parâmetro configurado no arquivo de definição de back-end.

Como faço para definir a criptografia de volume por meio do Trident com o ONTAP?

Você pode definir a criptografia no volume provisionado pelo Trident usando o parâmetro de criptografia no arquivo de definição de back-end. Para obter mais informações, consulte: "Como o Trident funciona com NVE e NAE"

Qual é a melhor maneira de implementar QoS para ONTAP por meio do Trident?

```
`StorageClasses`Use para implementar QoS para ONTAP.
```

Como especificar o provisionamento thin ou thick por meio do Trident?

Os drivers ONTAP oferecem suporte ao provisionamento thin ou thick. Os drivers do ONTAP são padrão para thin Provisioning. Se o provisionamento espesso for desejado, você deverá configurar o arquivo de definição de back-end ou o `StorageClass`. Se ambos estiverem configurados, `StorageClass` tem precedência. Configure o seguinte para o ONTAP:

1. On `StorageClass`, defina o `provisioningType` atributo como thick (espesso).
2. No arquivo de definição de back-end, ative volumes espessos definindo `backend spaceReserve parameter` como volume.

Como posso garantir que os volumes que estão a ser utilizados não sejam eliminados mesmo que elimine acidentalmente o PVC?

A proteção de PVC é ativada automaticamente no Kubernetes a partir da versão 1,10.

Posso expandir PVCs de NFS criados pela Trident?

Sim. Você pode expandir um PVC NFS criado pelo Trident. As Trident Autogrow Policies podem expandir automaticamente os PVCs com base em limites de capacidade. O dimensionamento automático de volume do ONTAP é separado e se aplica aos ONTAP SAN drivers compatíveis: ele aumenta o FlexVol de suporte sem alterar o tamanho do PVC. Para obter mais informações, consulte "Dimensionamento automático de volume do ONTAP".

Posso importar um volume enquanto estiver no modo de proteção de dados (DP) da SnapMirror ou offline?

A importação de volume falha se o volume externo estiver no modo DP ou estiver offline. Você recebe a seguinte mensagem de erro:

```
Error: could not import volume: volume import failed to get size of
volume: volume <name> was not found (400 Bad Request) command terminated
with exit code 1.
```

```
Make sure to remove the DP mode or put the volume online before importing
the volume.
```

Como a cota de recursos é traduzida para um cluster NetApp?

A cota de recursos de armazenamento do Kubernetes deve funcionar enquanto o armazenamento do NetApp tiver capacidade. Quando o storage do NetApp não consegue honrar as configurações de cota do Kubernetes devido à falta de capacidade, o Trident tenta provisionar, mas faz erros.

Posso criar instantâneos de volume usando o Trident?

Sim. A criação de snapshots de volume sob demanda e volumes persistentes a partir de snapshots é compatível com o Trident. Para criar PVS a partir de instantâneos, certifique-se de que a `VolumeSnapshotDataSource` porta de recurso foi ativada.

Quais são os drivers compatíveis com snapshots de volume Trident?

A partir de hoje, o suporte para snapshots sob demanda está disponível para nossos produtos. `ontap-nas` , `ontap-nas-flexgroup` , `ontap-san` , `ontap-san-economy` , `solidfire-san` , e `azure-netapp-files` Drivers de backend.

Como faço para fazer um backup instantâneo de um volume provisionado pelo Trident com o ONTAP?

Isso está disponível nos `ontap-nas` drivers , `ontap-san` e `ontap-nas-flexgroup` . Você também pode especificar um `snapshotPolicy` para o `ontap-san-economy` driver no nível FlexVol.

Isso também está disponível `ontap-nas-economy` nos drivers, mas na granularidade de nível FlexVol volume e não na granularidade de nível de qtree. Para habilitar a capacidade de snapshot volumes provisionados pelo Trident, defina a opção de parâmetro de back-end `snapshotPolicy` para a política de snapshot desejada, conforme definido no back-end do ONTAP. Todos os instantâneos obtidos pelo controlador de armazenamento não são conhecidos pelo Trident.

Posso definir uma porcentagem de reserva de snapshot para um volume provisionado por meio do Trident?

Sim, você pode reservar uma porcentagem específica de espaço em disco para armazenar as cópias snapshot através do Trident definindo o `snapshotReserve` atributo no arquivo de definição do back-end. Se você configurou `snapshotPolicy` e `snapshotReserve` no arquivo de definição de back-end, a porcentagem de reserva de snapshot é definida de acordo com a `snapshotReserve` porcentagem mencionada no arquivo de back-end. Se o `snapshotReserve` número percentual não for mencionado, ONTAP por padrão leva a porcentagem de reserva de snapshot como 5. Se a `snapshotPolicy` opção estiver definida como None (nenhum), a percentagem de reserva de instantâneos é definida como 0.

Posso acessar diretamente o diretório instantâneo do volume e copiar arquivos?

Sim, você pode acessar o diretório instantâneo no volume provisionado pelo Trident definindo o `snapshotDir` parâmetro no arquivo de definição de back-end.

Posso configurar o SnapMirror para volumes através do Trident?

Atualmente, o SnapMirror precisa ser definido externamente usando a CLI ou o OnCommand System Manager do ONTAP.

Como faço para restaurar volumes persistentes para um snapshot específico do ONTAP?

Para restaurar um volume para um instantâneo do ONTAP, execute as seguintes etapas:

1. Quiesce o pod do aplicativo que está usando o volume persistente.
2. Reverter para o snapshot necessário por meio da CLI ou OnCommand System Manager do ONTAP.
3. Reinicie o pod de aplicativos.

O Trident provisiona volumes em SVMs que têm um espelhamento de compartilhamento de carga configurado?

Os espelhos de compartilhamento de carga podem ser criados para volumes raiz de SVMs que fornecem dados por NFS. O ONTAP atualiza automaticamente os espelhos de compartilhamento de carga para volumes criados pelo Trident. Isso pode resultar em atrasos nos volumes de montagem. Quando vários volumes são criados usando o Trident, o provisionamento de um volume depende da atualização do espelhamento de compartilhamento de carga do ONTAP.

Como posso separar o uso da classe de storage para cada cliente/locatário?

O Kubernetes não permite classes de storage em namespaces. No entanto, você pode usar o Kubernetes para limitar o uso de uma classe de armazenamento específica por namespace usando cotas de recursos de armazenamento, que são por namespace. Para negar acesso a um namespace específico a um armazenamento específico, defina a cota de recurso como 0 para essa classe de armazenamento.

Solução de problemas

Use os ponteiros fornecidos aqui para solucionar problemas que você pode encontrar ao instalar e usar o Trident.

OBSERVAÇÃO	Para obter ajuda com o Trident, crie um pacote de suporte usando `tridentctl logs -a -n trident` e envie-o para o suporte da NetApp.

Resolução de problemas gerais

- Se o pod Trident não aparecer corretamente (por exemplo, quando o pod Trident está preso `ContainerCreating` na fase com menos de dois contêineres prontos), em execução `kubectl -n trident describe deployment trident` e `kubectl -n trident describe pod trident-**` pode fornecer informações adicionais. A obtenção de logs do kubelet (por exemplo, via `journalctl -xeu kubelet`) também pode ser útil.
- Se não houver informações suficientes nos logs do Trident, você pode tentar ativar o modo de depuração para o Trident passando o `-d` sinalizador para o parâmetro de instalação com base na opção de instalação.

  Em seguida, confirme se a depuração é definida usando `./tridentctl logs -n trident` e pesquisando `level=debug msg` no log.

  Instalado com Operador

  ```
  kubectl patch torc trident -n <namespace> --type=merge -p
  '{"spec":{"debug":true}}'
  ```

Isso reiniciará todos os pods do Trident, o que pode levar vários segundos. Você pode verificar isso observando a coluna 'IDADE' na saída do `kubectl get pod -n trident`.

Para Trident 20,07 e 20,10, utilizar `tprov` no lugar `torc` de .

Instalado com Helm

```
helm upgrade <name> trident-operator-21.07.1-custom.tgz --set
tridentDebug=true`
```

Instalado com tridentctl

```
./tridentctl uninstall -n trident
./tridentctl install -d -n trident
```

- Você também pode obter logs de depuração para cada back-end, incluindo `debugTraceFlags` na sua definição de back-end. Por exemplo, inclua `debugTraceFlags: {"api":true, "method":true,}` para obter chamadas de API e transversais de método nos logs do Trident. Os backends existentes podem ter `debugTraceFlags` sido configurados com um `tridentctl backend update`.
- Ao usar o Red Hat Enterprise Linux CoreOS (RHCOS), certifique-se de que `iscsid` está habilitado nos nós de trabalho e iniciado por padrão. Isso pode ser feito usando OpenShift MachineConfigs ou modificando os modelos de ignição.
- Um problema comum que você pode encontrar ao usar Trident "Azure NetApp Files" é quando os segredos do locatário e do cliente vêm de um registro de aplicativo com permissões insuficientes. Para obter uma lista completa dos requisitos do Trident, consulte a "Azure NetApp Files"configuração.
- Se houver problemas com a montagem de um PV em um recipiente, certifique-se de que `rpcbind` está instalado e funcionando. Use o gerenciador de pacotes necessário para o sistema operacional do host e verifique se `rpcbind` está em execução. Você pode verificar o status `rpcbind` do serviço executando um `systemctl status rpcbind` ou seu equivalente.
- Se um back-end do Trident relatar que ele está `failed` no estado apesar de ter trabalhado antes, provavelmente será causado pela alteração das credenciais do SVM/administrador associadas ao back-end. Atualizar as informações de back-end usando `tridentctl update backend` ou saltando o pod Trident corrigirá esse problema.
- Se você encontrar problemas de permissão ao instalar o Trident com Docker como o runtime do contentor, tente a instalação do Trident com o `--in cluster=false` sinalizador. Isso não usará um pod do instalador e evitará problemas de permissão vistos devido ao `trident-installer` usuário.
- Utilize o `uninstall parameter <Uninstalling Trident>` para limpar após uma falha de funcionamento. Por padrão, o script não remove os CRDs que foram criados pelo Trident, tornando seguro desinstalar e instalar novamente mesmo em uma implantação em execução.
- Se você quiser fazer o downgrade para uma versão anterior do Trident, primeiro execute o `tridentctl uninstall` comando para remover o Trident. Baixe o desejado "Versão Trident" e instale usando o `tridentctl install` comando.
- Após uma instalação bem-sucedida, se um PVC estiver preso na `Pending` fase, a execução `kubectl describe pvc` pode fornecer informações adicionais sobre por que a Trident não conseguiu fornecer um PV para este PVC.

Implantação sem êxito do Trident usando o operador

Se você estiver implantando o Trident usando o operador, o status das `TridentOrchestrator` alterações será alterado de `Installing` para `Installed`. Se você observar o `Failed` status e o operador não conseguir se recuperar sozinho, você deve verificar os logs do operador executando o seguinte comando:

```
tridentctl logs -l trident-operator
```

Arrastar os logs do contentor do operador Trident pode apontar para onde está o problema. Por exemplo, um desses problemas poderia ser a incapacidade de extrair as imagens de contentor necessárias de Registros upstream em um ambiente com airgapped.

Para entender por que a instalação do Trident não foi bem-sucedida, você deve dar uma olhada no `TridentOrchestrator` status.

```
kubectl describe torc trident-2
Name:         trident-2
Namespace:
Labels:       <none>
Annotations:  <none>
API Version:  trident.netapp.io/v1
Kind:         TridentOrchestrator
...
Status:
  Current Installation Params:
    IPv6:
    Autosupport Hostname:
    Autosupport Image:
    Autosupport Proxy:
    Autosupport Serial Number:
    Debug:
    Image Pull Secrets:         <nil>
    Image Registry:
    k8sTimeout:
    Kubelet Dir:
    Log Format:
    Silence Autosupport:
    Trident Image:
  Message:                      Trident is bound to another CR 'trident'
  Namespace:                    trident-2
  Status:                       Error
  Version:
Events:
  Type     Reason  Age                From                          Message
  ----     ------  ----               ----                          -------
  Warning  Error   16s (x2 over 16s)  trident-operator.netapp.io  Trident
```

```
is bound to another CR 'trident'
```

Este erro indica que já existe um `TridentOrchestrator` que foi usado para instalar o Trident. Como cada cluster do Kubernetes pode ter apenas uma instância do Trident, o operador garante que, a qualquer momento, só exista uma ativa `TridentOrchestrator` que possa criar.

Além disso, observar o status dos pods do Trident geralmente pode indicar se algo não está certo.

```
kubectl get pods -n trident

NAME                                READY   STATUS             RESTARTS
AGE
trident-csi-4p5kq                   1/2     ImagePullBackOff   0
5m18s
trident-csi-6f45bfd8b6-vfrkw        4/5     ImagePullBackOff   0
5m19s
trident-csi-9q5xc                   1/2     ImagePullBackOff   0
5m18s
trident-csi-9v95z                   1/2     ImagePullBackOff   0
5m18s
trident-operator-766f7b8658-ldzsv   1/1     Running            0
8m17s
```

Você pode ver claramente que os pods não são capazes de inicializar completamente porque uma ou mais imagens de contentor não foram obtidas.

Para resolver o problema, você deve editar o `TridentOrchestrator` CR. Alternativamente, você pode excluir `TridentOrchestrator` e criar um novo com a definição modificada e precisa.

Implantação sem êxito do Trident usando `tridentctl`

Para ajudar a descobrir o que deu errado, você pode executar o instalador novamente usando o `-d` argumento, que irá ativar o modo de depuração e ajudá-lo a entender qual é o problema:

```
./tridentctl install -n trident -d
```

Depois de resolver o problema, você pode limpar a instalação da seguinte forma e, em seguida, executar o `tridentctl install` comando novamente:

```
./tridentctl uninstall -n trident
INFO Deleted Trident deployment.
INFO Deleted cluster role binding.
INFO Deleted cluster role.
INFO Deleted service account.
INFO Removed Trident user from security context constraint.
```

```
INFO Trident uninstallation succeeded.
```

Remova completamente Trident e CRDs

Você pode remover completamente o Trident e todos os CRDs criados e recursos personalizados associados.

AVISO	Isso não pode ser desfeito. Não faça isso a menos que você queira uma instalação completamente nova do Trident. Para desinstalar o Trident sem remover CRDs, "Desinstale o Trident"consulte .

Operador Trident

Para desinstalar o Trident e remover completamente CRDs usando o operador Trident:

```
kubectl patch torc <trident-orchestrator-name> --type=merge -p
'{"spec":{"wipeout":["crds"],"uninstall":true}}'
```

Leme

Para desinstalar o Trident e remover completamente CRDs usando Helm:

```
kubectl patch torc trident --type=merge -p
'{"spec":{"wipeout":["crds"],"uninstall":true}}'
```

<code>dtridentctl</code>

Para remover completamente CRDs após desinstalar o Trident usando `tridentctl`

```
tridentctl obliviate crd
```

Falha de desinstalação do nó NVMe com namespaces de bloco bruto RWX do Kubernetes 1,26

Se você estiver executando o Kubernetes 1,26, a desinstalação de nós pode falhar ao usar NVMe/TCP com namespaces de bloco bruto RWX. Os cenários a seguir fornecem uma solução para a falha. Como alternativa, você pode atualizar o Kubernetes para 1,27.

Excluiu o namespace e o pod

Considere um cenário em que você tenha um namespace gerenciado do Trident (volume persistente NVMe) anexado a um pod. Se você excluir o namespace diretamente do back-end do ONTAP, o processo de despreparo fica preso após tentar excluir o pod. Esse cenário não afeta o cluster do Kubernetes ou outras funcionalidades.

Solução alternativa

Desmonte o volume persistente (correspondente a esse namespace) do respetivo nó e exclua-o.

Dados bloqueados

```
 If you block (or bring down) all the dataLIFs of the NVMe Trident
backend, the unstaging process gets stuck when you attempt to delete the
pod. In this scenario, you cannot run any NVMe CLI commands on the
Kubernetes node.
.Solução alternativa
Abra o dataLIFS para restaurar a funcionalidade completa.
```

Mapeamento de namespace excluído

```
 If you remove the `hostNQN` of the worker node from the corresponding
subsystem, the unstaging process gets stuck when you attempt to delete the
pod. In this scenario, you cannot run any NVMe CLI commands on the
Kubernetes node.
.Solução alternativa
Adicione o `hostNQN` de volta ao subsistema.
```

Os clientes NFSv4.2 relatam "argumento inválido" após atualizar o ONTAP quando esperavam que "v4.2-xattrs" estivesse habilitado

Após a atualização do ONTAP, os clientes NFSv4.2 podem relatar erros de "argumento inválido" ao tentar montar exportações NFSv4.2. Este problema ocorre quando o `v4.2-xattrs` a opção não está habilitada no SVM. .Solução alternativa Habilite o `v4.2-xattrs` opção no SVM ou atualize para o ONTAP 9.12.1 ou posterior, onde esta opção é habilitada por padrão.

Suporte

O NetApp fornece suporte para Trident de várias maneiras. Amplas opções gratuitas de suporte autônomo estão disponíveis 24 horas por dia, 7 dias por semana, como artigos da base de conhecimento (KB) e um canal discord.

Ciclo de vida do suporte da Trident

O Trident oferece três níveis de suporte com base na sua versão. "Suporte à versão do software NetApp para definições"Consulte a .

Suporte completo
: O Trident fornece suporte completo por doze meses a partir da data de lançamento.

Suporte limitado
: O Trident fornece suporte limitado durante os meses de 13 a 24 a partir da data de lançamento.

Auto-suporte
: A documentação do Trident está disponível para os meses 25 - 36 a partir da data de lançamento.

Versão	Fim do suporte total	Fim do suporte limitado	Fim do autoatendimento
"26,10"	Outubro de 2027	Outubro de 2028	Outubro 2029
"26,06"	Julho de 2027	Julho de 2028	Julho de 2029
"26,02"	Fevereiro de 2027	Fevereiro de 2028	Fevereiro de 2029
"25,10"	Outubro de 2026	Outubro de 2027	Outubro de 2028
"25,06"	Junho de 2026	Junho de 2027	Junho de 2028
"25,02"	Fevereiro de 2026	Fevereiro de 2027	Fevereiro de 2028
"24,10"	—	Outubro de 2026	Outubro de 2027
"24,06"	—	Junho de 2026	Junho de 2027
"24,02"	—	Fevereiro de 2026	Fevereiro de 2027
"23,10"	—	—	Outubro de 2026
"23,07"	—	—	Julho de 2026
"23,04"	—	—	Abril de 2026

Auto-suporte

Para obter uma lista completa de artigos de resolução de problemas, consulte "Base de Conhecimento NetApp (login necessário)".

Apoio comunitário

Há uma vibrante comunidade pública de usuários de contentores (incluindo desenvolvedores do Trident) no "Canal discord"nosso . Este é um ótimo lugar para fazer perguntas gerais sobre o projeto e discutir tópicos relacionados com colegas de mentalidade semelhante.

Suporte técnico da NetApp

Para obter ajuda com o Trident, crie um pacote de suporte usando `tridentctl logs -a -n trident` e envie-o para `NetApp Support <Getting Help>`o .

Para mais informações

- "Recursos do Trident"
- "Kubernetes Hub"

Referência

Portas Trident

Saiba mais sobre as portas que o Trident usa para comunicação.

Visão geral

Trident utiliza diversas portas para comunicação dentro de clusters Kubernetes e com storage backends. A seguir, um resumo das principais portas, suas finalidades e considerações de segurança.

- **Foco em tráfego de saída**: Os nós do Kubernetes (controller e worker) iniciam principalmente o tráfego para LIFs/IPs de storage, portanto, as regras do iptables devem permitir tráfego de saída dos IPs dos nós para IPs de storage específicos nessas portas. Evite regras amplas do tipo "qualquer para qualquer".
- **Restrições de entrada**: limite as portas internas do Trident ao tráfego interno do cluster (por exemplo, usando CNI como Calico). Nenhuma exposição desnecessária de entrada nos firewalls do host.
- **Segurança de protocolo**:
  - Use TCP sempre que possível (mais confiável).
  - Habilite CHAP/IPsec para iSCSI se sensível; TLS/HTTPS para gerenciamento (porta 443/8443).
  - Para NFSv4 (padrão no Trident), remova as portas UDP/NFSv3 mais antigas (por exemplo, 4045-4049) se não forem necessárias.
  - Restrinja o acesso a sub-redes confiáveis; monitore com ferramentas como Prometheus (porta 8001 opcional).

Portas para nós controladores

Essas portas são destinadas principalmente ao Trident operator (gerenciamento de backend). Todas as portas internas são de nível de pod; permita nos nós somente se o firewall do host interferir com o CNI.

Porta/Protocolo	Direção	Finalidade	Driver/Protocolo	Notas de segurança
TCP 8000	Entrada/Saída (cluster-internal)	Servidor REST Trident (comunicação operador-controlador)	Tudo	Restringir aos CIDRs dos pods; sem exposição externa.
TCP 8443	Entrada/Saída (cluster-internal)	Backchannel HTTPS (API interna segura)	Tudo	Criptografia TLS; limite ao service mesh do Kubernetes se utilizado.
TCP 8001	Entrada (interna ao cluster, opcional)	Métricas Prometheus	Tudo	Exponha apenas às ferramentas de monitoramento (por exemplo, usando RBAC); desative se não for usado.
TCP 443	Saída	HTTPS para ONTAP SVM/cluster mgmt LIF	ONTAP (todos), ANF	Exigir validação de certificado TLS; restringir apenas aos IPs de mgmt do LIF.

Porta/Protocolo	Direção	Finalidade	Driver/Protocolo	Notas de segurança
TCP 8443	Saída	HTTPS para Proxy de Serviços Web E-Series	E-Series (iSCSI)	API REST padrão; utiliza certificados; configurável no YAML do backend.

Portas para nós de trabalho

Essas portas são para daemonsets de nós CSI e montagens de pods. As portas de dados são de saída para LIFs de dados de storage; inclua extras do NFSv3 se estiver usando NFSv3 (opcional para NFSv4).

Porta/Protocolo	Direção	Finalidade	Driver/Protocolo	Notas de segurança
TCP 17546	Entrada (local para pod)	Sondas de liveness/prontidão do nó CSI	Tudo	Configurável (--probe-port); garanta que não haja conflitos de host; somente local.
TCP 8000	Entrada/Saída (cluster-internal)	SERVIDOR REST do Trident	Tudo	Como acima; interno ao pod.
TCP 8443	Entrada/Saída (cluster-internal)	Backchannel HTTPS	Tudo	Como acima.
TCP 8001	Entrada (interna ao cluster, opcional)	Métricas Prometheus	Tudo	Como acima.
TCP 443	Saída	HTTPS para ONTAP SVM/cluster mgmt LIF	ONTAP (todos), ANF	Conforme mencionado acima; usado para descoberta.
TCP 8443	Saída	HTTPS para Proxy de Serviços Web E-Series	E-Series (iSCSI)	Como acima.
TCP/UDP 111	Saída	RPCBIND/portmapper	ONTAP-NAS (NFSv3/v4), ANF (NFS)	Obrigatório para v3; opcional para v4 (descarregamento do firewall); restringir se estiver usando somente NFSv4.
TCP/UDP 2049	Saída	Daemon NFS	ONTAP-NAS (NFSv3/v4), ANF (NFS)	Dados essenciais; bem conhecidos; use TCP para confiabilidade.
TCP/UDP 635	Saída	Daemon de montagem	ONTAP-NAS (NFSv3/v4), ANF (NFS)	Montagem; callbacks bidirecionais possíveis (permitir entrada efêmera se necessário).
UDP 4045	Saída	Gerenciador de bloqueio NFS (nlockmgr)	ONTAP-NAS (NFSv3)	Bloqueio de arquivos; ignorar para v4 (pNFS handles); somente UDP.

Porta/Protocolo	Direção	Finalidade	Driver/Protocolo	Notas de segurança
UDP 4046	Saída	Monitor de status NFS (statd)	ONTAP-NAS (NFSv3)	Notificações; podem ser necessárias portas efêmeras de entrada (1024-65535) para callbacks.
UDP 4049	Saída	Daemon de cotas NFS (rquotad)	ONTAP-NAS (NFSv3)	Cotas; pular para v4.
TCP 3260	Saída	destino iSCSI (descoberta/dados/CHAP)	ONTAP-SAN (iSCSI), E-Series (iSCSI)	Bem conhecido; autenticação CHAP nesta porta; habilite CHAP mútuo para segurança.
TCP 445	Saída	SMB/CIFS	ONTAP-NAS (SMB), ANF (SMB)	Bem conhecido; use SMB3 com criptografia (Trident annotation netapp.io/smb-encryption=true).
TCP/UDP 88 (opcional)	Saída	Autenticação Kerberos	ONTAP (NFS/SMB/iSCSI com Kerberos)	Se estiver usando Kerberos (não é o padrão); para servidores AD, não para storage.
TCP/UDP 389 (opcional)	Saída	LDAP	ONTAP (NFS/SMB com LDAP)	Semelhante; para resolução de nomes/autenticação; restringir ao AD.

OBSERVAÇÃO A porta da sonda de disponibilidade/disponibilidade pode ser alterada durante a instalação utilizando o `--probe-port` sinalizador. É importante garantir que essa porta não esteja sendo usada por outro processo nos nós de trabalho.

API REST do Trident

Embora "comandos e opções tridentctl" sejam a maneira mais fácil de interagir com a Trident API REST, você pode usar o endpoint REST diretamente se preferir.

Quando usar a API REST

A API REST é útil para instalações avançadas que usam o Trident como um binário autônomo em implantações não Kubernetes.

Para uma melhor segurança, o Trident `REST API` é restrito ao localhost por padrão ao ser executado dentro de um pod. Para alterar esse comportamento, você precisa definir o argumento do Trident `-address` em sua configuração de pod.

Usando a API REST

Para exemplos de como essas APIs são chamadas, passe o (`-d`sinalizador debug). Para obter mais informações, "Gerenciar o Trident usando o tridentctl"consulte .

A API funciona da seguinte forma:

OBTER

`GET <trident-address>/trident/v1/<object-type>`

Lista todos os objetos desse tipo.

`GET <trident-address>/trident/v1/<object-type>/<object-name>`

Obtém os detalhes do objeto nomeado.

POST

`POST <trident-address>/trident/v1/<object-type>`

Cria um objeto do tipo especificado.

- Requer uma configuração JSON para o objeto a ser criado. Para obter a especificação de cada tipo de objeto, "Gerenciar o Trident usando o tridentctl"consulte a .
- Se o objeto já existir, o comportamento varia: Os backends atualizam o objeto existente, enquanto todos os outros tipos de objeto falharão a operação.

ELIMINAR

`DELETE <trident-address>/trident/v1/<object-type>/<object-name>`

Exclui o recurso nomeado.

OBSERVAÇÃO	Os volumes associados a backends ou classes de armazenamento continuarão a existir; estes devem ser excluídos separadamente. Para obter mais informações, "Gerenciar o Trident usando o tridentctl"consulte .

Opções de linha de comando

O Trident expõe várias opções de linha de comando para o orquestrador Trident. Você pode usar essas opções para modificar sua implantação.

A registar

`-debug`

Ativa a saída de depuração.

`-loglevel <level>`

Define o nível de log (debug, info, warn, error, fatal). O padrão é INFO.

Kubernetes

`-k8s_pod`

Use essa opção ou `-k8s_api_server` para ativar o suporte do Kubernetes. Isso faz com que o Trident use suas credenciais de conta de serviço do Kubernetes do pod que contêm para entrar em Contato com o servidor de API. Isso só funciona quando o Trident é executado como um pod em um cluster do Kubernetes com contas de serviço ativadas.

-k8s_api_server <insecure-address:insecure-port>

Use essa opção ou `-k8s_pod` para ativar o suporte do Kubernetes. Quando especificado, o Trident se coneta ao servidor de API do Kubernetes usando o endereço e a porta inseguros fornecidos. Isso permite que o Trident seja implantado fora de um pod; no entanto, ele só suporta conexões inseguras com o servidor de API. Para se conetar com segurança, implante o Trident em um pod com a `-k8s_pod` opção.

Docker

-volume_driver <name>

Nome do driver usado ao Registrar o plug-in do Docker. O padrão é `netapp`.

-driver_port <port-number>

Ouça nesta porta em vez de um soquete de domínio UNIX.

-config <file>

Obrigatório; você deve especificar esse caminho para um arquivo de configuração de back-end.

DESCANSO

-address <ip-or-host>

Especifica o endereço no qual o servidor REST do Trident deve ouvir. O padrão é localhost. Ao ouvir no localhost e executar dentro de um pod Kubernetes, a interface REST não é diretamente acessível de fora do pod. `-address ""`Utilize para tornar a INTERFACE REST acessível a partir do endereço IP do pod.

CUIDADO	A interface REST DO Trident pode ser configurada para ouvir e servir apenas em 127.0.0.1 (para IPv4) ou [::1] (para IPv6).

-port <port-number>

Especifica a porta na qual o servidor REST do Trident deve ouvir. O padrão é 8000.

-rest

Ativa a interface REST. O padrão é verdadeiro.

Objetos Kubernetes e Trident

É possível interagir com o Kubernetes e o Trident usando APIs REST lendo e escrevendo objetos de recursos. Há vários objetos de recursos que ditam a relação entre o Kubernetes e o Trident, o Trident e o storage, o Kubernetes e o storage. Alguns desses objetos são gerenciados pelo Kubernetes e os outros são gerenciados pelo Trident.

Como os objetos interagem uns com os outros?

Talvez a maneira mais fácil de entender os objetos, para que eles são e como eles interagem seja seguir uma única solicitação de armazenamento de um usuário do Kubernetes:

1. Um usuário cria `PersistentVolumeClaim` uma solicitação de um novo `PersistentVolume` de um tamanho específico de um Kubernetes `StorageClass` que foi configurado anteriormente pelo administrador.

2. O Kubernetes `StorageClass` identifica o Trident como seu provisionador e inclui parâmetros que informam ao Trident como provisionar um volume para a classe solicitada.
3. O Trident olha para si `StorageClass` mesmo com o mesmo nome que identifica a correspondência `Backends` e `StoragePools` que pode usar para provisionar volumes para a classe.
4. O Trident provisiona o storage em um back-end compatível e cria dois objetos: Um `PersistentVolume` no Kubernetes que diz ao Kubernetes como encontrar, montar e tratar o volume e um volume no Trident que mantém a relação entre o `PersistentVolume` e o storage real.
5. O Kubernetes vincula `PersistentVolumeClaim` o ao novo `PersistentVolume`. Pods que incluem a `PersistentVolumeClaim` montagem que Persistentvolume em qualquer host em que ele seja executado.
6. Um usuário cria um `VolumeSnapshot` de um PVC existente, usando um `VolumeSnapshotClass` que aponta para Trident.
7. O Trident identifica o volume que está associado ao PVC e cria um snapshot do volume em seu back-end. Ele também cria um `VolumeSnapshotContent` que instrui o Kubernetes sobre como identificar o snapshot.
8. Um usuário pode criar um `PersistentVolumeClaim` usando `VolumeSnapshot` como fonte.
9. O Trident identifica o instantâneo necessário e executa o mesmo conjunto de etapas envolvidas na criação de um `PersistentVolume` e um `Volume`.

DICA Para ler mais sobre objetos do Kubernetes, é altamente recomendável que você leia a "Volumes persistentes" seção da documentação do Kubernetes.

Objetos do Kubernetes `PersistentVolumeClaim`

Um objeto Kubernetes `PersistentVolumeClaim` é uma solicitação de storage feita por um usuário de cluster do Kubernetes.

Além da especificação padrão, o Trident permite que os usuários especifiquem as seguintes anotações específicas de volume se quiserem substituir os padrões definidos na configuração de back-end:

Anotação	Opção de volume	Drivers suportados
Trident.NetApp.io/sistema de arquivos	Sistema de ficheiros	ONTAP-san, SolidFire-san, ONTAP-san-economy
Trident.NetApp.io/cloneFromPVC	CloneSourcevolume	ontap-nas, ontap-san, solidfire-san, azure-netapp-files, ontap-san-economy
Trident.NetApp.io/splitOnClone	SplitOnClone	ONTAP-nas, ONTAP-san
Trident.NetApp.io/protocolo	protocolo	qualquer
Trident.NetApp.io/exportPolicy	Política de exportação	ONTAP-nas, ONTAP-nas-economy, ONTAP-nas-FlexGroup
Trident.NetApp.io/snapshotPolicy	Política de SnapshotPolicy	ONTAP-nas, ONTAP-nas-economy, ONTAP-nas-FlexGroup, ONTAP-san

Anotação	Opção de volume	Drivers suportados
Trident.NetApp.io/snapshotServe	SnapshotServe	ontap-nas, ontap-nas-flexgroup, ontap-san
Trident.NetApp.io/snapshotDirectory	SnapshotDirectory	ONTAP-nas, ONTAP-nas-economy, ONTAP-nas-FlexGroup
Trident.NetApp.io/unixPermissions	UnixPermissions	ONTAP-nas, ONTAP-nas-economy, ONTAP-nas-FlexGroup
Trident.NetApp.io/blocksize	Tamanho do bloco	SolidFire-san
trident.netapp.io/skipRecoveryQueue	pularFilaDeRecuperação	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, ontap-san-economy

Se o PV criado tiver a `Delete` política de recuperação, o Trident excluirá o PV e o volume de backup quando o PV for liberado (ou seja, quando o usuário exclui o PVC). Caso a ação de exclusão falhe, o Trident marca o PV como tal e periodicamente tenta novamente a operação até que seja bem-sucedida ou o PV seja excluído manualmente. Se o PV usar a `Retain` política, o Trident a ignora e assume que o administrador irá limpá-la do Kubernetes e do back-end, permitindo que o volume seja feito backup ou inspecionado antes de sua remoção. Observe que a exclusão do PV não faz com que o Trident exclua o volume de backup. Você deve removê-lo usando a API REST (`tridentctl`).

O Trident dá suporte à criação de snapshots de volume usando a especificação CSI: Você pode criar um instantâneo de volume e usá-lo como fonte de dados para clonar PVCs existentes. Dessa forma, cópias pontuais de PVS podem ser expostas ao Kubernetes na forma de snapshots. Os instantâneos podem então ser usados para criar novos PVS. Dê uma olhada `On-Demand Volume Snapshots` para ver como isso funcionaria.

O Trident também fornece as `cloneFromPVC` anotações e `splitOnClone` para a criação de clones. Você pode usar essas anotações para clonar um PVC sem precisar usar a implementação do CSI.

Aqui está um exemplo: Se um usuário já tem um PVC chamado `mysql`, o usuário pode criar um novo PVC chamado `mysqlclone` usando a anotação, como `trident.netapp.io/cloneFromPVC: mysql` . Com esse conjunto de anotações, o Trident clona o volume correspondente ao PVC mysql, em vez de provisionar um volume do zero.

Considere os seguintes pontos:

- A NetApp recomenda clonar um volume ocioso.
- O PVC e seu clone devem estar no mesmo namespace do Kubernetes e ter a mesma classe de storage.
- Com os `ontap-nas` drivers e `ontap-san`, pode ser desejável definir a anotação de PVC `trident.netapp.io/splitOnClone` em conjunto `trident.netapp.io/cloneFromPVC` com o . Com `trident.netapp.io/splitOnClone` definido como `true`, o Trident divide o volume clonado do volume pai e, portanto, desacoplando completamente o ciclo de vida do volume clonado de seus pais às custas de perder alguma eficiência de storage. Não `trident.netapp.io/splitOnClone` configurá-lo ou configurá-lo para `false` resultar em consumo de espaço reduzido no back-end à custa de criar dependências entre os volumes pai e clone, de modo que o volume pai não possa ser excluído a menos que o clone seja excluído primeiro. Um cenário em que dividir o clone faz sentido é clonar um volume de banco de dados vazio, onde é esperado que o volume e seu clone diverjam muito e não se beneficiem das eficiências de armazenamento oferecidas pelo ONTAP.

O `sample-input` diretório contém exemplos de definições de PVC para uso com Trident. Consulte a para

obter uma descrição completa dos parâmetros e definições associados aos volumes Trident.

Objetos do Kubernetes `PersistentVolume`

Um objeto Kubernetes `PersistentVolume` representa um storage disponibilizado para o cluster do Kubernetes. Ele tem um ciclo de vida que é independente do pod que o usa.

OBSERVAÇÃO	O Trident cria `PersistentVolume` objetos e os Registra no cluster do Kubernetes automaticamente com base nos volumes provisionados. Você não é esperado para gerenciá-los sozinho.

Quando você cria um PVC que se refere a um Trident-based `StorageClass`, o Trident provisiona um novo volume usando a classe de armazenamento correspondente e Registra um novo PV para esse volume. Ao configurar o volume provisionado e o PV correspondente, o Trident segue as seguintes regras:

- O Trident gera um nome PV para o Kubernetes e um nome interno que ele usa para provisionar o storage. Em ambos os casos, é garantir que os nomes são únicos em seu escopo.
- O tamanho do volume corresponde ao tamanho solicitado no PVC o mais próximo possível, embora possa ser arredondado para a quantidade alocável mais próxima, dependendo da plataforma.

Objetos do Kubernetes `StorageClass`

Os objetos Kubernetes `StorageClass` são especificados por nome em `PersistentVolumeClaims` para provisionar o storage com um conjunto de propriedades. A própria classe de storage identifica o provisionador a ser usado e define esse conjunto de propriedades em termos que o provisionador entende.

É um dos dois objetos básicos que precisam ser criados e gerenciados pelo administrador. O outro é o objeto backend do Trident.

Um objeto do Kubernetes `StorageClass` que usa o Trident é parecido com este:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: <Name>
provisioner: csi.trident.netapp.io
mountOptions: <Mount Options>
parameters: <Trident Parameters>
allowVolumeExpansion: true
volumeBindingMode: Immediate
```

Esses parâmetros são específicos do Trident e informam à Trident como provisionar volumes para a classe.

Os parâmetros da classe de armazenamento são:

Atributo	Tipo	Obrigatório	Descrição
atributos	map[string]string	não	Veja a seção atributos abaixo

Atributo	Tipo	Obrigatório	Descrição
StoragePools	MAP[string]StringList	não	Mapa de nomes de back-end para listas de pools de armazenamento dentro
Além disso, StoragePools	MAP[string]StringList	não	Mapa de nomes de back-end para listas de pools de armazenamento dentro
Excluir StoragePools	MAP[string]StringList	não	Mapa de nomes de back-end para listas de pools de armazenamento dentro

Os atributos de storage e seus possíveis valores podem ser classificados em atributos de seleção de pool de storage e atributos do Kubernetes.

Atributos de seleção do pool de armazenamento

Esses parâmetros determinam quais pools de storage gerenciado pelo Trident devem ser utilizados para provisionar volumes de um determinado tipo.

Atributo	Tipo	Valores	Oferta	Pedido	Suportado por
1	cadeia de carateres	hdd, híbrido, ssd	Pool contém Mídia desse tipo; híbrido significa ambos	Tipo de material especificado	ONTAP-nas, ONTAP-nas-economy, ONTAP-nas-FlexGroup, ONTAP-san, SolidFire-san
ProvisioningType	cadeia de carateres	fino, grosso	O pool é compatível com esse método de provisionamento	Método de provisionamento especificado	thick: all ONTAP; thin: all ONTAP & SolidFire-san
BackendType	cadeia de carateres	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, solidfire-san, azure-netapp-files, ontap-san-economy	Pool pertence a este tipo de backend	Back-end especificado	Todos os drivers
instantâneos	bool	verdadeiro, falso	O pool é compatível com volumes com snapshots	Volume com instantâneos ativados	ontap-nas, ontap-san, solidfire-san
clones	bool	verdadeiro, falso	O pool é compatível com volumes de clonagem	Volume com clones ativados	ontap-nas, ontap-san, solidfire-san

Atributo	Tipo	Valores	Oferta	Pedido	Suportado por
criptografia	bool	verdadeiro, falso	O pool é compatível com volumes criptografados	Volume com encriptação ativada	ONTAP-nas, ONTAP-nas-economy, ONTAP-nas-flexgroups, ONTAP-san
IOPS	int	número inteiro positivo	O pool é capaz de garantir IOPS nessa faixa	Volume garantido estas operações de entrada/saída por segundo	SolidFire-san

1: Não suportado pelos sistemas ONTAP Select

Na maioria dos casos, os valores solicitados influenciam diretamente o provisionamento; por exemplo, a solicitação de provisionamento espesso resulta em um volume provisionado rapidamente. No entanto, um pool de storage de elemento usa o mínimo e o máximo de IOPS oferecidos para definir valores de QoS, em vez do valor solicitado. Nesse caso, o valor solicitado é usado apenas para selecionar o pool de armazenamento.

O ideal é usar `attributes` sozinho para modelar as qualidades do storage de que você precisa para atender às necessidades de uma classe específica. O Trident deteta e seleciona automaticamente pools de armazenamento que correspondem a *all* do `attributes` que você especificar.

Se você não conseguir usar `attributes` para selecionar automaticamente os pools certos para uma classe, use os `storagePools` parâmetros e `additionalStoragePools` para refinar ainda mais os pools ou até mesmo selecionar um conjunto específico de pools.

Você pode usar o `storagePools` parâmetro para restringir ainda mais o conjunto de pools que correspondem a qualquer `attributes` especificado . Em outras palavras, o Trident usa a interseção de pools identificados pelos `attributes` parâmetros e `storagePools` para o provisionamento. Você pode usar um parâmetro sozinho ou ambos juntos.

Você pode usar o `additionalStoragePools` parâmetro para estender o conjunto de pools que o Trident usa para provisionamento, independentemente de quaisquer pools selecionados pelos `attributes` parâmetros e. `storagePools`

Você pode usar o `excludeStoragePools` parâmetro para filtrar o conjunto de pools que o Trident usa para provisionar. O uso desse parâmetro remove todos os pools que correspondem.

```
`storagePools`Nos parâmetros e `additionalStoragePools`, cada entrada
assume o formulário `<backend>:<storagePoolList>`, onde
`<storagePoolList>` é uma lista separada por vírgulas de pools de
armazenamento para o back-end especificado. Por exemplo, um valor para
`additionalStoragePools` pode parecer como
`ontapnas_192.168.1.100:aggr1,aggr2;solidfire_192.168.1.101:bronze`. Essas
listas aceitam valores de regex tanto para os valores de backend quanto de
lista. Você pode usar `tridentctl get backend` para obter a lista de
```

```
backends e suas piscinas.
```

Atributos do Kubernetes

Esses atributos não têm impacto na seleção de pools de storage/back-ends pelo Trident durante o provisionamento dinâmico. Em vez disso, esses atributos simplesmente fornecem parâmetros compatíveis com volumes persistentes do Kubernetes. Os nós de trabalho são responsáveis pelas operações de criação de sistema de arquivos e podem exigir utilitários de sistema de arquivos, como xfsprogs.

Atributo	Tipo	Valores	Descrição	Drivers relevantes	Versão do Kubernetes
FsType	cadeia de carateres	ext4, ext3, xfs	O tipo de sistema de arquivos para volumes de bloco	SolidFire-san, ONTAP-nas, ONTAP-nas-economy, ONTAP-nas-FlexGroup, ONTAP-san, ONTAP-san-economy	Tudo
AllowVolumeExpansion	booleano	verdadeiro, falso	Ative ou desative o suporte para aumentar o tamanho do PVC	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, ontap-san-economy, solidfire-san, azure-netapp-files	Mais de 1,11 anos
VolumeBindingMode	cadeia de carateres	Imediato, WaitForFirstConsumer	Escolha quando ocorre a vinculação de volume e o provisionamento dinâmico	Tudo	1,19 - 1,26

DICA

- O `fsType` parâmetro é usado para controlar o tipo de sistema de arquivos desejado para LUNs SAN. Além disso, o Kubernetes também usa a presença de `fsType` em uma classe de armazenamento para indicar que existe um sistema de arquivos. A propriedade do volume só pode ser controlada usando o `fsGroup` contexto de segurança de um pod se `fsType` estiver definido. "Kubernetes: Configurar um contexto de segurança para um pod ou contêiner"Consulte para obter uma visão geral sobre como definir a propriedade do volume usando o `fsGroup` contexto. O Kubernetes aplicará o `fsGroup` valor somente se:
  - `fsType` é definido na classe de armazenamento.
  - O modo de acesso de PVC é RWO.

  Para drivers de armazenamento NFS, já existe um sistema de arquivos como parte da exportação NFS. Para usar `fsGroup` a classe de armazenamento ainda precisa

especificar um `fsType`. você pode configurá-lo como `nfs` ou qualquer valor não nulo.

- "Expanda volumes"Consulte para obter mais detalhes sobre a expansão do volume.
- O pacote de instalação do Trident fornece vários exemplos de definições de classe de armazenamento para uso com o Trident no `sample-input/storage-class-*.yaml`. A exclusão de uma classe de armazenamento Kubernetes faz com que a classe de armazenamento Trident correspondente também seja excluída.

Objetos do Kubernetes `VolumeSnapshotClass`

Os objetos do Kubernetes `VolumeSnapshotClass` são análogos ao `StorageClasses`. Eles ajudam a definir várias classes de armazenamento e são referenciados por instantâneos de volume para associar o snapshot à classe de snapshot necessária. Cada snapshot de volume é associado a uma classe de snapshot de volume único.

A `VolumeSnapshotClass` deve ser definida por um administrador para criar instantâneos. Uma classe de instantâneo de volume é criada com a seguinte definição:

```
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshotClass
metadata:
  name: csi-snapclass
driver: csi.trident.netapp.io
deletionPolicy: Delete
```

O `driver` especifica ao Kubernetes que as solicitações de snapshots de volume `csi-snapclass` da classe são tratadas pelo Trident. O `deletionPolicy` especifica a ação a ser tomada quando um instantâneo deve ser excluído. `deletionPolicy`Quando está definido como `Delete`, os objetos instantâneos de volume e o instantâneo subjacente no cluster de armazenamento são removidos quando um instantâneo é excluído. Alternativamente, configurá-lo para `Retain` significa que `VolumeSnapshotContent` e o instantâneo físico são retidos.

Objetos do Kubernetes `VolumeSnapshot`

Um objeto Kubernetes `VolumeSnapshot` é uma solicitação para criar um snapshot de um volume. Assim como um PVC representa uma solicitação feita por um usuário para um volume, um instantâneo de volume é uma solicitação feita por um usuário para criar um instantâneo de um PVC existente.

Quando uma solicitação de snapshot de volume entra, o Trident gerencia automaticamente a criação do snapshot para o volume no back-end e expõe o snapshot criando um objeto exclusivo `VolumeSnapshotContent`. Você pode criar snapshots a partir de PVCs existentes e usar os snapshots como DataSource ao criar novos PVCs.

OBSERVAÇÃO	O ciclo de vida de um VolumeSnapshot é independente do PVC de origem: um snapshot persiste mesmo após o PVC de origem ser excluído. Ao excluir um PVC que tenha instantâneos associados, o Trident marca o volume de apoio para este PVC em um estado **Deletando**, mas não o remove completamente. O volume é removido quando todos os instantâneos associados são excluídos.

Objetos do Kubernetes `VolumeSnapshotContent`

Um objeto Kubernetes `VolumeSnapshotContent` representa um snapshot retirado de um volume já provisionado. Ele é análogo a `PersistentVolume` e significa um snapshot provisionado no cluster de storage. Semelhante aos `PersistentVolumeClaim` objetos e `PersistentVolume`, quando um snapshot é criado, o `VolumeSnapshotContent` objeto mantém um mapeamento um-para-um para o `VolumeSnapshot` objeto, que havia solicitado a criação do snapshot.

O `VolumeSnapshotContent` objeto contém detalhes que identificam exclusivamente o instantâneo, como o `snapshotHandle`. Esta `snapshotHandle` é uma combinação única do nome do PV e do nome do `VolumeSnapshotContent` objeto.

Quando uma solicitação de snapshot entra, o Trident cria o snapshot no back-end. Depois que o snapshot é criado, o Trident configura um `VolumeSnapshotContent` objeto e, portanto, expõe o snapshot à API do Kubernetes.

OBSERVAÇÃO	Normalmente, você não precisa gerenciar o `VolumeSnapshotContent` objeto. Uma exceção a isso é quando você deseja "importar um instantâneo de volume"criar fora do Trident.

Objetos do Kubernetes `VolumeGroupSnapshotClass`

Os objetos do Kubernetes `VolumeGroupSnapshotClass` são análogos ao `VolumeSnapshotClass`. Eles ajudam a definir várias classes de armazenamento e são referenciados por snapshots de grupos de volumes para associar o snapshot à classe de snapshot necessária. Cada snapshot de grupo de volumes é associado a uma única classe de snapshot de grupo de volumes.

UM `VolumeGroupSnapshotClass` deve ser definido por um administrador para criar um grupo de snapshots. Uma classe de snapshot de grupo de volumes é criada com a seguinte definição:

```
apiVersion: groupsnapshot.storage.k8s.io/v1beta1
kind: VolumeGroupSnapshotClass
metadata:
  name: csi-group-snap-class
  annotations:
    kubernetes.io/description: "Trident group snapshot class"
driver: csi.trident.netapp.io
deletionPolicy: Delete
```

O `driver` especifica ao Kubernetes que as solicitações de instantâneos do grupo de volumes do `csi-group-snap-class` classe são gerenciados pelo Trident. O `deletionPolicy` especifica a ação a ser tomada quando um instantâneo de grupo deve ser excluído. Quando `deletionPolicy` está definido para `Delete` , os objetos de instantâneo do grupo de volumes, bem como o instantâneo subjacente no cluster de armazenamento, são removidos quando um instantâneo é excluído. Alternativamente, configurá-lo para `Retain` significa que `VolumeGroupSnapshotContent` e o instantâneo físico são retidos.

Objetos do Kubernetes `VolumeGroupSnapshot`

Um Kubernetes `VolumeGroupSnapshot` objeto é uma solicitação para criar um snapshot de vários volumes.

Assim como um PVC representa uma solicitação feita por um usuário para um volume, um snapshot de grupo de volumes é uma solicitação feita por um usuário para criar um snapshot de um PVC existente.

Quando uma solicitação de instantâneo de grupo de volume chega, o Trident gerencia automaticamente a criação do instantâneo de grupo para os volumes no backend e expõe o instantâneo criando um único `VolumeGroupSnapshotContent` objeto. Você pode criar snapshots a partir de PVCs existentes e usar os snapshots como DataSource ao criar novos PVCs.

OBSERVAÇÃO	O ciclo de vida de um VolumeGroupSnapshot é independente do PVC de origem: um snapshot persiste mesmo após a exclusão do PVC de origem. Ao excluir um PVC que tenha instantâneos associados, o Trident marca o volume de apoio para este PVC em um estado **Deletando**, mas não o remove completamente. O snapshot do grupo de volumes é removido quando todos os snapshots associados são excluídos.

Objetos do Kubernetes `VolumeGroupSnapshotContent`

Um Kubernetes `VolumeGroupSnapshotContent` objeto representa um instantâneo de grupo obtido de um volume já provisionado. Ele é análogo a `PersistentVolume` e significa um snapshot provisionado no cluster de storage. Semelhante aos `PersistentVolumeClaim` objetos e `PersistentVolume`, quando um snapshot é criado, o `VolumeSnapshotContent` objeto mantém um mapeamento um-para-um para o `VolumeSnapshot` objeto, que havia solicitado a criação do snapshot.

O `VolumeGroupSnapshotContent` objeto contém detalhes que identificam o grupo de instantâneos, como `volumeGroupSnapshotHandle` e volumeSnapshotHandles individuais existentes no sistema de armazenamento.

Quando uma solicitação de snapshot é recebida, o Trident cria o snapshot do grupo de volumes no backend. Após a criação do snapshot do grupo de volumes, o Trident configura um `VolumeGroupSnapshotContent` objeto e, assim, expõe o instantâneo à API do Kubernetes.

Objetos do Kubernetes `CustomResourceDefinition`

Os recursos personalizados do Kubernetes são endpoints na API do Kubernetes que são definidos pelo administrador e são usados para agrupar objetos semelhantes. O Kubernetes dá suporte à criação de recursos personalizados para armazenar uma coleção de objetos. Você pode obter essas definições de recursos executando `kubectl get crds`o .

As definições personalizadas de recursos (CRDs) e os metadados de objetos associados são armazenados pelo Kubernetes em seu armazenamento de metadados. Isso elimina a necessidade de uma loja separada para o Trident.

O Trident usa `CustomResourceDefinition` objetos para preservar a identidade de objetos do Trident, como backends Trident, classes de storage Trident e volumes Trident. Esses objetos são gerenciados pelo Trident. Além disso, a estrutura de snapshot do volume CSI introduz algumas CRDs que são necessárias para definir snapshots de volume.

CRDs são uma construção do Kubernetes. Os objetos dos recursos definidos acima são criados pelo Trident. Como um exemplo simples, quando um back-end é criado usando `tridentctl`o , um objeto CRD correspondente `tridentbackends` é criado para consumo pelo Kubernetes.

Aqui estão alguns pontos a ter em mente sobre os CRDs do Trident:

- Quando o Trident é instalado, um conjunto de CRDs é criado e pode ser usado como qualquer outro tipo de recurso.
- Ao desinstalar o Trident usando o `tridentctl uninstall` comando, os pods Trident são excluídos, mas os CRDs criados não são limpos. "Desinstale o Trident"Consulte para compreender como o Trident pode ser completamente removido e reconfigurado do zero.

ObjetosTrident `StorageClass`

O Trident cria classes de storage correspondentes para objetos Kubernetes `StorageClass` que especificam `csi.trident.netapp.io` no campo do provisionador. O nome da classe de storage corresponde ao do objeto Kubernetes `StorageClass` que ele representa.

OBSERVAÇÃO	Com o Kubernetes, esses objetos são criados automaticamente quando um Kubernetes `StorageClass` que usa o Trident como provisionador é registrado.

As classes de armazenamento compreendem um conjunto de requisitos para volumes. O Trident atende a esses requisitos com os atributos presentes em cada pool de storage. Se forem correspondentes, esse pool de storage será um destino válido para volumes de provisionamento que usam essa classe de storage.

Você pode criar configurações de classe de armazenamento para definir diretamente classes de armazenamento usando a API REST. No entanto, para implantações do Kubernetes, esperamos que elas sejam criadas ao Registrar novos objetos do Kubernetes `StorageClass`.

Objetos de back-end do Trident

Os backends representam os fornecedores de storage em cima dos quais o Trident provisiona volumes. Uma única instância do Trident pode gerenciar qualquer número de backends.

OBSERVAÇÃO	Este é um dos dois tipos de objetos que você cria e gerencia a si mesmo. O outro é o objeto Kubernetes `StorageClass`.

Para obter mais informações sobre como construir esses objetos, "configurando backends"consulte .

ObjetosTrident `StoragePool`

Os pools de armazenamento representam os locais distintos disponíveis para provisionamento em cada backend. Para o ONTAP, estes correspondem a agregados em SVMs. Para NetApp HCI/ SolidFire, estes correspondem às bandas de QoS especificadas pelo administrador. Cada pool de armazenamento possui um conjunto de atributos de armazenamento distintos, que definem suas características de desempenho e de proteção de dados.

Ao contrário dos outros objetos aqui, os candidatos ao pool de armazenamento são sempre descobertos e gerenciados automaticamente.

ObjetosTrident `Volume`

Os volumes são a unidade básica de provisionamento, incluindo pontos de extremidade de back-end, como compartilhamentos NFS e iSCSI e FC LUNs. No Kubernetes, eles correspondem diretamente `PersistentVolumes` ao . Ao criar um volume, certifique-se de que ele tenha uma classe de armazenamento, que determina onde esse volume pode ser provisionado, juntamente com um tamanho.

OBSERVAÇÃO

- No Kubernetes, esses objetos são gerenciados automaticamente. Você pode visualizá-los para ver o que o Trident provisionou.
- Ao excluir um PV com instantâneos associados, o volume Trident correspondente é atualizado para um estado **Deletando**. Para que o volume Trident seja excluído, você deve remover os snapshots do volume.

Uma configuração de volume define as propriedades que um volume provisionado deve ter.

Atributo	Tipo	Obrigatório	Descrição
versão	cadeia de carateres	não	Versão da API Trident ("1")
nome	cadeia de carateres	sim	Nome do volume a criar
StorageClass	cadeia de carateres	sim	Classe de storage a ser usada ao provisionar o volume
tamanho	cadeia de carateres	sim	Tamanho do volume a provisionar em bytes
protocolo	cadeia de carateres	não	Tipo de protocolo a utilizar; "ficheiro" ou "bloco"
InternalName	cadeia de carateres	não	Nome do objeto no sistema de storage; gerado pelo Trident
CloneSourcevolume	cadeia de carateres	não	ONTAP (nas, san) & SolidFire-*: Nome do volume a partir do qual clonar
SplitOnClone	cadeia de carateres	não	ONTAP (nas, san): Divida o clone de seu pai
Política de SnapshotPolicy	cadeia de carateres	não	ONTAP-*: Política de snapshot a ser usada
SnapshotServe	cadeia de carateres	não	ONTAP-*: Porcentagem de volume reservado para snapshots
Política de exportação	cadeia de carateres	não	ONTAP-nas*: Política de exportação para usar
SnapshotDirectory	bool	não	ONTAP-nas*: Se o diretório snapshot está visível
UnixPermissions	cadeia de carateres	não	ONTAP-nas*: Permissões iniciais do UNIX
Tamanho do bloco	cadeia de carateres	não	SolidFire-*: Tamanho do bloco/setor

Atributo	Tipo	Obrigatório	Descrição
Sistema de ficheiros	cadeia de carateres	não	Tipo de sistema de ficheiros
pularFilaDeRecuperação	cadeia de carateres	não	Durante a exclusão de um volume, ignore a fila de recuperação no armazenamento e exclua o volume imediatamente.

O Trident gera `internalName` ao criar o volume. Isto consiste em duas etapas. Primeiro, ele prepende o prefixo de armazenamento (o padrão `trident` ou o prefixo na configuração de back-end) para o nome do volume, resultando em um nome do formulário `<prefix>-<volume-name>`. Em seguida, procede à higienização do nome, substituindo carateres não permitidos no backend. Para backends ONTAP, ele substitui hífens por sublinhados (assim, o nome interno se torna `<prefix>_<volume-name>`). Para backends de elemento, ele substitui sublinhados por hífens.

Você pode usar configurações de volume para provisionar volumes diretamente usando a API REST, mas nas implantações do Kubernetes, esperamos que a maioria dos usuários use o método padrão do Kubernetes `PersistentVolumeClaim`. O Trident cria esse objeto de volume automaticamente como parte do processo de provisionamento.

ObjetosTrident `Snapshot`

Os snapshots são uma cópia pontual de volumes, que pode ser usada para provisionar novos volumes ou restaurar o estado. No Kubernetes, eles correspondem diretamente a `VolumeSnapshotContent` objetos. Cada snapshot é associado a um volume, que é a origem dos dados do snapshot.

Cada `Snapshot` objeto inclui as propriedades listadas abaixo:

Atributo	Tipo	Obrigatório	Descrição
versão	Cadeia de carateres	Sim	Versão da API Trident ("1")
nome	Cadeia de carateres	Sim	Nome do objeto snapshot Trident
InternalName	Cadeia de carateres	Sim	Nome do objeto snapshot Trident no sistema de storage
Nome do volume	Cadeia de carateres	Sim	Nome do volume persistente para o qual o instantâneo é criado
VolumeInternalName	Cadeia de carateres	Sim	Nome do objeto de volume Trident associado no sistema de storage

OBSERVAÇÃO	No Kubernetes, esses objetos são gerenciados automaticamente. Você pode visualizá-los para ver o que o Trident provisionou.

Quando uma solicitação de objeto Kubernetes `VolumeSnapshot` é criada, o Trident funciona criando um

objeto snapshot no sistema de storage de backup. `internalName`O deste objeto instantâneo é gerado combinando o prefixo `snapshot-` com o `UID` do `VolumeSnapshot` objeto (por exemplo, `snapshot-e8d8a0ca-9826-11e9-9807-525400f3f660`). `volumeName` e `volumeInternalName` são preenchidos obtendo os detalhes do volume de apoio.

Objeto Trident `ResourceQuota`

O deamonset do Trident consome uma `system-node-critical` classe de prioridade - a classe de prioridade mais alta disponível no Kubernetes - para garantir que o Trident possa identificar e limpar volumes durante o desligamento gracioso do nó e permitir que os pods do Trident daemonset pré-empt cargas de trabalho com prioridade mais baixa em clusters onde há alta pressão de recursos.

Para conseguir isso, o Trident emprega um `ResourceQuota` objeto para garantir que uma classe de prioridade "system-node-critical" no daemonset do Trident esteja satisfeita. Antes da implantação e criação do daemonset, o Trident procura o `ResourceQuota` objeto e, se não for descoberto, o aplica.

Se você precisar de mais controle sobre a cota de recurso padrão e Classe de prioridade, você pode gerar um `custom.yaml` ou configurar o `ResourceQuota` objeto usando o gráfico de Helm.

O seguinte é um exemplo de um objeto 'ResourceQuota' priorizando o daemonset do Trident.

```
apiVersion: <version>
kind: ResourceQuota
metadata:
  name: trident-csi
  labels:
    app: node.csi.trident.netapp.io
spec:
  scopeSelector:
    matchExpressions:
      - operator: In
        scopeName: PriorityClass
        values:
          - system-node-critical
```

Para obter mais informações sobre cotas de recursos, "Kubernetes: Cotas de recursos"consulte .

Limpe `ResourceQuota` se a instalação falhar

No caso raro em que a instalação falha depois que o `ResourceQuota` objeto é criado, primeiro "desinstalação"tente e depois reinstale.

Se isso não funcionar, remova manualmente o `ResourceQuota` objeto.

Retire `ResourceQuota`

Se você preferir controlar sua própria alocação de recursos, você pode remover o objeto Trident `ResourceQuota` usando o comando:

```
kubectl delete quota trident-csi -n trident
```

Padrões de segurança do pod (PSS) e restrições de contexto de segurança (SCC)

Os padrões de segurança do pod do Kubernetes (PSS) e as políticas de segurança do Pod (PSP) definem níveis de permissão e restringem o comportamento dos pods. As restrições de contexto de Segurança OpenShift (SCC) definem similarmente a restrição de pod específica ao OpenShift Kubernetes Engine. Para fornecer essa personalização, o Trident permite certas permissões durante a instalação. As seções a seguir detalham as permissões definidas pelo Trident.

OBSERVAÇÃO	O PSS substitui as políticas de segurança do Pod (PSP). A PSP foi obsoleta no Kubernetes v1,21 e será removida em v1,25. Para obter mais informações, "Kubernetes: Segurança"consulte .

Contexto de segurança do Kubernetes necessário e campos relacionados

Permissão	Descrição
Privilegiado	O CSI exige que os pontos de montagem sejam bidirecionais, o que significa que o pod de nó do Trident deve executar um contentor privilegiado. Para obter mais informações, "Kubernetes: Propagação de montagem"consulte .
Rede de host	Necessário para o daemon iSCSI. `iscsiadm` Gerencia montagens iSCSI e usa redes de host para se comunicar com o daemon iSCSI.
IPC do host	O NFS usa comunicação entre processos (IPC) para se comunicar com o NFSD.
PID do host	Necessário para iniciar `rpc-statd` o NFS. O Trident consulta os processos de host para determinar se `rpc-statd` está sendo executado antes da montagem de volumes NFS.

Recursos	O `SYS_ADMIN` recurso é fornecido como parte dos recursos padrão para contentores privilegiados. Por exemplo, o Docker define esses recursos para contentores privilegiados:
	`CapPrm: 0000003fffffffff`
	`CapEff: 0000003fffffffff`

Seccomp	O perfil Seccomp é sempre "unconfinado" em contentores privilegiados; portanto, não pode ser habilitado no Trident.

Permissão	Descrição
SELinux	No OpenShift, os contentores privilegiados são executados no `spc_t` domínio ("contentor Super privilegiado") e os contentores sem privilégios são executados `container_t` no domínio. No `containerd`, com `container-selinux` instalado, todos os contentores são executados no `spc_t` domínio, o que desativa efetivamente o SELinux. Portanto, o Trident não adiciona `seLinuxOptions` a contêineres.
DAC	Os contentores privilegiados devem ser executados como root. Os contentores não privilegiados são executados como root para acessar os sockets unix exigidos pelo CSI.

Padrões de segurança do pod (PSS)

Etiqueta	Descrição	Padrão
`pod-security.kubernetes.io/enforce` `pod-security.kubernetes.io/enforce-version`	Permite que o controlador Trident e os nós sejam admitidos no namespace de instalação. Não altere a etiqueta do namespace.	`enforce: privileged enforce-version: <version of the current cluster or highest version of PSS tested.>`

AVISO Alterar os rótulos do namespace pode resultar em pods não sendo programados, um "erro ao criar: …" ou, "Aviso: Trident-csi-…". Se isso acontecer, verifique se a etiqueta do namespace para `privileged` foi alterada. Em caso afirmativo, reinstale o Trident.

Políticas de segurança do pod (PSP)

Campo	Descrição	Padrão
`allowPrivilegeEscalation`	Os contêineres privilegiados devem permitir o escalonamento de privilégios.	`true`
`allowedCSIDrivers`	O Trident não usa volumes efêmeros de CSI inline.	Vazio
`allowedCapabilities`	Os contêineres Trident não privilegiados não exigem mais recursos do que o conjunto padrão e os contentores privilegiados recebem todos os recursos possíveis.	Vazio
`allowedFlexVolumes`	O Trident não faz uso de um "Controlador Flexvolume", portanto, eles não estão incluídos na lista de volumes permitidos.	Vazio

Campo	Descrição	Padrão
`allowedHostPaths`	O pod de nó Trident monta o sistema de arquivos raiz do nó, portanto, não há benefício para definir esta lista.	Vazio
`allowedProcMountTypes`	O Trident não usa nenhum `ProcMountTypes`.	Vazio
`allowedUnsafeSysctls`	O Trident não requer nenhum inseguro `sysctls`.	Vazio
`defaultAddCapabilities`	Não são necessários recursos para serem adicionados a contentores privilegiados.	Vazio
`defaultAllowPrivilegeEscalation`	Permitir o escalonamento de privilégios é Tratado em cada pod Trident.	`false`
`forbiddenSysctls`	Não `sysctls` são permitidos.	Vazio
`fsGroup`	Os contêineres do Trident são executados como raiz.	`RunAsAny`
`hostIPC`	A montagem de volumes NFS requer que o IPC do host se comunique com `nfsd`	`true`
`hostNetwork`	O iscsiadm requer que a rede host se comunique com o daemon iSCSI.	`true`
`hostPID`	O PID do host é necessário para verificar se `rpc-statd` está sendo executado no nó.	`true`
`hostPorts`	O Trident não usa nenhuma porta de host.	Vazio
`privileged`	Os pods de nós do Trident devem executar um contêiner privilegiado para montar volumes.	`true`
`readOnlyRootFilesystem`	Os pods de nós do Trident devem gravar no sistema de arquivos do nó.	`false`
`requiredDropCapabilities`	Os pods de nós do Trident executam um contêiner privilegiado e não podem descartar recursos.	`none`
`runAsGroup`	Os contêineres do Trident são executados como raiz.	`RunAsAny`
`runAsUser`	Os contêineres do Trident são executados como raiz.	`runAsAny`
`runtimeClass`	O Trident não usa `RuntimeClasses`o .	Vazio

Campo	Descrição	Padrão
`seLinux`	O Trident não define `seLinuxOptions` porque atualmente existem diferenças em como os tempos de execução de contêineres e as distribuições do Kubernetes lidam com o SELinux.	Vazio
`supplementalGroups`	Os contêineres do Trident são executados como raiz.	`RunAsAny`
`volumes`	Os pods do Trident exigem esses plugins de volume.	`hostPath, projected, emptyDir`

Restrições de contexto de segurança (SCC)

Etiquetas	Descrição	Padrão
`allowHostDirVolumePlugin`	Os pods de nó Trident montam o sistema de arquivos raiz do nó.	`true`
`allowHostIPC`	A montagem de volumes NFS requer que o IPC do host se comunique com `nfsd`o .	`true`
`allowHostNetwork`	O iscsiadm requer que a rede host se comunique com o daemon iSCSI.	`true`
`allowHostPID`	O PID do host é necessário para verificar se `rpc-statd` está sendo executado no nó.	`true`
`allowHostPorts`	O Trident não usa nenhuma porta de host.	`false`
`allowPrivilegeEscalation`	Os contêineres privilegiados devem permitir o escalonamento de privilégios.	`true`
`allowPrivilegedContainer`	Os pods de nós do Trident devem executar um contêiner privilegiado para montar volumes.	`true`
`allowedUnsafeSysctls`	O Trident não requer nenhum inseguro `sysctls`.	`none`
`allowedCapabilities`	Os contêineres Trident não privilegiados não exigem mais recursos do que o conjunto padrão e os contentores privilegiados recebem todos os recursos possíveis.	Vazio
`defaultAddCapabilities`	Não são necessários recursos para serem adicionados a contentores privilegiados.	Vazio

Etiquetas	Descrição	Padrão
`fsGroup`	Os contêineres do Trident são executados como raiz.	`RunAsAny`
`groups`	Este SCC é específico do Trident e está vinculado ao seu usuário.	Vazio
`readOnlyRootFilesystem`	Os pods de nós do Trident devem gravar no sistema de arquivos do nó.	`false`
`requiredDropCapabilities`	Os pods de nós do Trident executam um contêiner privilegiado e não podem descartar recursos.	`none`
`runAsUser`	Os contêineres do Trident são executados como raiz.	`RunAsAny`
`seLinuxContext`	O Trident não define `seLinuxOptions` porque atualmente existem diferenças em como os tempos de execução de contêineres e as distribuições do Kubernetes lidam com o SELinux.	Vazio
`seccompProfiles`	Os contentores privilegiados funcionam sempre "sem confinamentos".	Vazio
`supplementalGroups`	Os contêineres do Trident são executados como raiz.	`RunAsAny`
`users`	Uma entrada é fornecida para vincular esse SCC ao usuário Trident no namespace Trident.	n/a.
`volumes`	Os pods do Trident exigem esses plugins de volume.	`hostPath, downwardAPI, projected, emptyDir`

Avisos legais

Avisos legais fornecem acesso a declarações de direitos autorais, marcas registradas, patentes e muito mais.

Direitos de autor

"https://www.netapp.com/company/legal/copyright/"

Marcas comerciais

NetApp, o logotipo DA NetApp e as marcas listadas na página de marcas comerciais da NetApp são marcas comerciais da NetApp, Inc. Outros nomes de produtos e empresas podem ser marcas comerciais de seus respetivos proprietários.

"https://www.netapp.com/company/legal/trademarks/"

Patentes

Uma lista atual de patentes de propriedade da NetApp pode ser encontrada em:

https://www.netapp.com/pdf.html?item=/media/11887-patentspage.pdf

Política de privacidade

"https://www.netapp.com/company/legal/privacy-policy/"

Código aberto

Você pode revisar os direitos autorais e as licenças de terceiros usados no software NetApp para Trident no arquivo de avisos para cada versão em https://github.com/NetApp/trident/.

Informações sobre direitos autorais

Copyright © 2026 NetApp, Inc. Todos os direitos reservados. Impresso nos EUA. Nenhuma parte deste documento protegida por direitos autorais pode ser reproduzida de qualquer forma ou por qualquer meio — gráfico, eletrônico ou mecânico, incluindo fotocópia, gravação, gravação em fita ou storage em um sistema de recuperação eletrônica — sem permissão prévia, por escrito, do proprietário dos direitos autorais.

O software derivado do material da NetApp protegido por direitos autorais está sujeito à seguinte licença e isenção de responsabilidade:

ESTE SOFTWARE É FORNECIDO PELA NETAPP "NO PRESENTE ESTADO" E SEM QUAISQUER GARANTIAS EXPRESSAS OU IMPLÍCITAS, INCLUINDO, SEM LIMITAÇÕES, GARANTIAS IMPLÍCITAS DE COMERCIALIZAÇÃO E ADEQUAÇÃO A UM DETERMINADO PROPÓSITO, CONFORME A ISENÇÃO DE RESPONSABILIDADE DESTE DOCUMENTO. EM HIPÓTESE ALGUMA A NETAPP SERÁ RESPONSÁVEL POR QUALQUER DANO DIRETO, INDIRETO, INCIDENTAL, ESPECIAL, EXEMPLAR OU CONSEQUENCIAL (INCLUINDO, SEM LIMITAÇÕES, AQUISIÇÃO DE PRODUTOS OU SERVIÇOS SOBRESSALENTES; PERDA DE USO, DADOS OU LUCROS; OU INTERRUPÇÃO DOS NEGÓCIOS), INDEPENDENTEMENTE DA CAUSA E DO PRINCÍPIO DE RESPONSABILIDADE, SEJA EM CONTRATO, POR RESPONSABILIDADE OBJETIVA OU PREJUÍZO (INCLUINDO NEGLIGÊNCIA OU DE OUTRO MODO), RESULTANTE DO USO DESTE SOFTWARE, MESMO SE ADVERTIDA DA RESPONSABILIDADE DE TAL DANO.

A NetApp reserva-se o direito de alterar quaisquer produtos descritos neste documento, a qualquer momento e sem aviso. A NetApp não assume nenhuma responsabilidade nem obrigação decorrentes do uso dos produtos descritos neste documento, exceto conforme expressamente acordado por escrito pela NetApp. O uso ou a compra deste produto não representam uma licença sob quaisquer direitos de patente, direitos de marca comercial ou quaisquer outros direitos de propriedade intelectual da NetApp.

O produto descrito neste manual pode estar protegido por uma ou mais patentes dos EUA, patentes estrangeiras ou pedidos pendentes.

LEGENDA DE DIREITOS LIMITADOS: o uso, a duplicação ou a divulgação pelo governo estão sujeitos a restrições conforme estabelecido no subparágrafo (b)(3) dos Direitos em Dados Técnicos - Itens Não Comerciais no DFARS 252.227-7013 (fevereiro de 2014) e no FAR 52.227- 19 (dezembro de 2007).

Os dados aqui contidos pertencem a um produto comercial e/ou serviço comercial (conforme definido no FAR 2.101) e são de propriedade da NetApp, Inc. Todos os dados técnicos e software de computador da NetApp fornecidos sob este Contrato são de natureza comercial e desenvolvidos exclusivamente com despesas privadas. O Governo dos EUA tem uma licença mundial limitada, irrevogável, não exclusiva, intransferível e não sublicenciável para usar os Dados que estão relacionados apenas com o suporte e para cumprir os contratos governamentais desse país que determinam o fornecimento de tais Dados. Salvo disposição em contrário no presente documento, não é permitido usar, divulgar, reproduzir, modificar, executar ou exibir os dados sem a aprovação prévia por escrito da NetApp, Inc. Os direitos de licença pertencentes ao governo dos Estados Unidos para o Departamento de Defesa estão limitados aos direitos identificados na cláusula 252.227-7015(b) (fevereiro de 2014) do DFARS.

Informações sobre marcas comerciais

NETAPP, o logotipo NETAPP e as marcas listadas em http://www.netapp.com/TM são marcas comerciais da NetApp, Inc. Outros nomes de produtos e empresas podem ser marcas comerciais de seus respectivos proprietários.