Proteja aplicativos com Trident Protect

Trident

NetApp
July 01, 2026

Índice

Proteja aplicativos com Trident Protect

Saiba mais sobre Trident Protect

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 pelos sistemas de armazenamento NetApp ONTAP e pelo provisionador de armazenamento NetApp Trident CSI. 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 Trident Protect criando recursos personalizados (CRs) ou usando a Trident Protect CLI.

Qual é o próximo passo?

Você pode conhecer os requisitos do Trident Protect antes de instalá-lo:

- "Requisitos do Trident Protect"

Instalar Trident Protect

Requisitos do Trident Protect

Comece verificando a prontidão do seu ambiente operacional, clusters de aplicativos, aplicativos e licenças. Certifique-se de que seu ambiente atenda a esses requisitos para implantar e operar Trident Protect.

Compatibilidade do cluster Kubernetes com o Trident Protect

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
- VMware Tanzu Portfolio
- Kubernetes upstream

- Os backups do Trident Protect são suportados apenas em nós de computação Linux. Nós de computação Windows não são suportados para operações de backup.
- Certifique-se de que o cluster no qual você instala Trident Protect esteja configurado com um controlador de snapshot em execução e os CRDs relacionados. Para instalar um controlador de snapshot, consulte "estas instruções".
- Certifique-se de que exista pelo menos um VolumeSnapshotClass. Para obter mais informações, consulte "VolumeSnapshotClass".
- É necessário o Helm 4.x ou posterior para instalar Trident Protect.

Compatibilidade do backend de armazenamento Trident Protect

Trident Protect é compatível com os seguintes storage backends:

- Amazon FSx for NetApp ONTAP
- Cloud Volumes ONTAP
- Matrizes de armazenamento ONTAP
- Google Cloud NetApp Volumes
- Azure NetApp Files

Certifique-se de que seu storage backend atenda aos seguintes requisitos:

- Certifique-se de que o armazenamento NetApp conectado ao cluster esteja usando Trident 24.02 ou mais recente (Trident 24.10 é recomendado).
- Certifique-se de ter um backend de storage NetApp ONTAP.
- Certifique-se de ter configurado um storage 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. 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-economy

Trident Protect oferece suporte a operações de backup e restauração em volumes nas-economy. Snapshots, clones e SnapMirror replicação para volumes nas-economy não são suportados atualmente. Você precisa habilitar um diretório de snapshot para cada volume nas-economy que planeja usar com Trident Protect.

Algumas aplicações não são compatíveis com volumes que utilizam um diretório de snapshot. Para essas aplicações, você precisa ocultar o diretório de snapshot executando o seguinte comando no sistema de storage ONTAP:

```
nfs modify -vserver <svm> -v3-hide-snapshot enabled
```

Você pode habilitar o diretório de snapshots executando o seguinte comando para cada volume nas-economy, substituindo `<volume-UUID>` pelo UUID do volume que deseja alterar:

```
tridentctl update volume <volume-UUID> --snapshot-dir=true --pool-level
=true -n trident
```

Você pode habilitar diretórios de snapshots por padrão para novos volumes definindo a opção de configuração do backend Trident `snapshotDir` para `true`. Volumes existentes não são afetados.

Protegendo dados com KubeVirt VMs

Trident Protect oferece recursos de congelamento e descongelamento do sistema de arquivos para máquinas virtuais KubeVirt durante operações de proteção de dados para garantir a consistência de 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 Helm chart.

Durante as operações de restauração, quaisquer `VirtualMachineSnapshots` criados para uma máquina virtual (VM) não são restaurados.

Trident Protect 25.10 e mais recentes

Trident Protect congela e descongela automaticamente os sistemas de arquivos 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 parâmetro `vm.freeze` durante a instalação do Helm chart. O parâmetro está ativado por padrão.

```
helm install ... --set vm.freeze=false ...
```

Trident Protect 24.10.1 a 25.06

A partir do Trident Protect 24.10.1, o Trident Protect congela e descongela automaticamente os sistemas de arquivos 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

Trident Protect 24.10 não garante automaticamente um estado consistente para os sistemas de arquivos de KubeVirt VM durante as operações de proteção de dados. Se você deseja proteger os dados da sua KubeVirt VM usando 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 "configurando virtualização" e, em seguida, usar o seguinte comando:

```
kubectl set env deployment/trident-protect-controller-manager
NEPTUNE_VM_FREEZE=true -n trident-protect
```

Requisitos para SnapMirror replication

NetApp SnapMirror a replicação está disponível para uso com 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 for NetApp ONTAP

Requisitos do cluster ONTAP para replicação SnapMirror

Certifique-se de que seu cluster ONTAP atenda aos seguintes requisitos caso planeje usar a replicação SnapMirror:

- **NetApp Trident**: NetApp Trident deve existir em ambos os clusters Kubernetes de origem e destino que utilizam ONTAP como backend. Trident Protect oferece suporte à replicação com a tecnologia NetApp SnapMirror usando classes de armazenamento suportadas pelos seguintes drivers:
  - `ontap-nas`: NFS
  - `ontap-san`: iSCSI
  - `ontap-san`: FC
  - `ontap-san`: NVMe/TCP (requer versão mínima do ONTAP 9.15.1)
- **Licenças**: As licenças assíncronas do ONTAP SnapMirror usando o pacote Data Protection devem estar habilitadas tanto nos clusters ONTAP de origem quanto de destino. Consulte "SnapMirror visão geral do licenciamento no ONTAP" para mais informações.

  A partir do ONTAP 9.10.1, todas as licenças são fornecidas como um arquivo de licença NetApp (NLF), que é um único arquivo que habilita vários recursos. Consulte "Licenças incluídas com o ONTAP One" para mais informações.

  Apenas a proteção assíncrona SnapMirror é suportada.

Considerações de peering para replicação SnapMirror

Certifique-se de que seu ambiente atenda aos seguintes requisitos caso planeje usar o peering de storage backend:

- **Cluster e SVM**: Os backends de armazenamento ONTAP devem estar emparelhados. Consulte "Visão geral do peering de cluster e SVM" para mais informações.

  Certifique-se de que os nomes SVM usados na relação de replicação entre dois clusters ONTAP sejam únicos.

- **NetApp Trident e SVM**: Os SVMs remotos emparelhados devem estar disponíveis para 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 do Trident / ONTAP para replicação SnapMirror

Trident Protect exige que você configure pelo menos um storage backend que suporte replicação para ambos os clusters de origem e destino. Se os clusters de origem e destino forem os mesmos, o aplicativo de destino deve usar um storage backend diferente do aplicativo de origem para a melhor resiliência.

Requisitos do cluster Kubernetes para SnapMirror replication

Certifique-se de que seus clusters Kubernetes atendam aos seguintes requisitos:

- **AppVault acessibilidade**: Tanto o cluster de origem quanto o cluster de destino devem ter acesso à rede para ler e gravar na AppVault para replicação de objetos da aplicação.
- **Conectividade de rede**: Configure regras de firewall, permissões de bucket e listas de permissão de IP para habilitar a comunicação entre ambos os clusters e o AppVault através das WANs.

  Muitos ambientes corporativos implementam políticas de firewall rigorosas em conexões WAN. Verifique esses requisitos de rede com sua equipe de infraestrutura antes de configurar a replicação.

Instalar e configurar Trident Protect

Se o seu ambiente atender aos requisitos do Trident Protect, você pode seguir estas etapas para instalar o Trident Protect no seu cluster. Você pode obter o Trident Protect da NetApp ou instalá-lo a partir do seu próprio registro privado. Instalar a partir de um registro privado é útil se o seu cluster não puder acessar a Internet.

Instalar Trident Protect

Instale Trident Protect a partir de NetApp

Passos

1. Adicione o repositório Trident:

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

2. Use o Helm para instalar Trident Protect. Substitua `<name-of-cluster>` pelo nome do 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 100.2602.0 --create
-namespace --namespace trident-protect
```

3. Opcionalmente, para ativar o registro de depuração (recomendado para solução de problemas), use:

```
helm install trident-protect netapp-trident-protect/trident-protect
--set clusterName=<name-of-cluster> --set logLevel=debug --version
100.2602.0 --create-namespace --namespace trident-protect
```

   O registro de depuração ajuda NetApp support a solucionar problemas sem exigir alterações no nível de registro ou reprodução do problema.

Instale Trident Protect a partir de um registro privado

Você pode instalar 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 pelas informações do seu ambiente:

Passos

1. Baixe as seguintes imagens para sua máquina local, atualize as tags e, em seguida, envie-as para seu registro privado:

```
docker.io/netapp/controller:26.02.0
docker.io/netapp/restic:26.02.0
docker.io/netapp/kopia:26.02.0
docker.io/netapp/kopiablockrestore:26.02.0
docker.io/netapp/trident-autosupport:26.02.0
docker.io/netapp/exechook:26.02.0
docker.io/netapp/resourcebackup:26.02.0
docker.io/netapp/resourcerestore:26.02.0
docker.io/netapp/resourcedelete:26.02.0
docker.io/netapp/trident-protect-utils:v1.0.0
```

Por exemplo:

```
docker pull docker.io/netapp/controller:26.02.0
```

```
docker tag docker.io/netapp/controller:26.02.0 <private-registry-url>/controller:26.02.0
```

```
docker push <private-registry-url>/controller:26.02.0
```

> Para obter o Helm chart, primeiro baixe o Helm chart em uma máquina com acesso à internet usando `helm pull trident-protect --version 100.2602.0 --repo https://netapp.github.io/trident-protect-helm-chart`, depois copie o arquivo resultante `trident-protect-100.2602.0.tgz` para seu ambiente offline e instale usando `helm install trident-protect ./trident-protect-100.2602.0.tgz` em vez da referência ao repositório na etapa final.

2. Crie o namespace do sistema Trident Protect:

```
kubectl create ns trident-protect
```

3. Faça login no registro:

```
helm registry login <private-registry-url> -u <account-id> -p <api-token>
```

4. Crie um segredo de pull para usar na autenticação do 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 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 ele contenha as seguintes configurações do Trident Protect:

```
---
imageRegistry: <private-registry-url>
imagePullSecrets:
  - name: regcred
```

> Os `imageRegistry` e `imagePullSecrets` valores se aplicam a todas as imagens de componentes, incluindo `resourcebackup` e `resourcerestore`. Se você enviar imagens para um caminho de repositório específico em seu registro (por exemplo, `example.com:443/my-repo`), inclua o caminho completo no campo de registro. Isso garantirá que todas as imagens sejam obtidas de `<private-registry-url>/<image-name>:<tag>`.

7. Use o Helm para instalar Trident Protect. Substitua `<name_of_cluster>` pelo nome do 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 100.2602.0 --create
-namespace --namespace trident-protect -f protectValues.yaml
```

8. Opcionalmente, para ativar o registro de depuração (recomendado para solução de problemas), use:

```
helm install trident-protect netapp-trident-protect/trident-protect
--set clusterName=<name-of-cluster> --set logLevel=debug --version
100.2602.0 --create-namespace --namespace trident-protect -f
protectValues.yaml
```

   O registro de depuração ajuda NetApp support a solucionar problemas sem exigir alterações no nível de registro ou reprodução do problema.

> Para opções adicionais de configuração do gráfico Helm, incluindo configurações do AutoSupport e filtragem de namespace, consulte "Personalizar 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 utilitário Trident `tridentctl`, 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 que usa para acessar seu cluster. Siga estas etapas, dependendo se sua máquina usa uma CPU x64 ou ARM.

Baixar 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/26.02.0/tridentctl-protect-linux-amd64
```

Baixar 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/26.02.0/tridentctl-protect-linux-arm64
```

Baixar 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/26.02.0/tridentctl-protect-macos-amd64
```

Baixar 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/26.02.0/tridentctl-protect-macos-arm64
```

1. Habilite as permissões de execução para o binário do plugin:

```
chmod +x tridentctl-protect
```

2. Copie o binário do plugin para um local definido na sua variável PATH. Por exemplo, `/usr/bin` ou `/usr/local/bin` (você pode precisar de privilégios elevados):

```
cp ./tridentctl-protect /usr/local/bin/
```

3. Opcionalmente, você pode copiar o binário do plugin para um local no seu diretório base. Nesse caso, é recomendável garantir que o local faça parte da sua variável PATH:

```
cp ./tridentctl-protect ~/bin/
```

Copiar o plugin para um local na sua variável PATH permite que você utilize o plugin digitando `tridentctl-protect` ou `tridentctl protect` de qualquer local.

Veja a ajuda do plugin Trident CLI

Você pode usar os recursos de ajuda incorporado do plugin para obter ajuda detalhada sobre as funcionalidades do plugin:

Passos

1. Use a função de ajuda para visualizar as instruções de utilização:

```
tridentctl-protect help
```

Ativar o preenchimento automático de comandos

Após instalar o plugin Trident Protect CLI, você pode ativar o recurso de autocompletar para determinados comandos.

Ative o recurso de autocompletar 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 base para conter o script:

```
mkdir -p ~/.bash/completions
```

3. Mova o script baixado para o diretório `~/.bash/completions`:

```
mv tridentctl-completion.bash ~/.bash/completions/
```

4. Adicione a seguinte linha ao arquivo `~/.bashrc` no seu diretório base:

```
source ~/.bash/completions/tridentctl-completion.bash
```

Ative o recurso de autocompletar para o Z shell

Passos

1. Crie o script de conclusão:

```
tridentctl-protect completion zsh > tridentctl-completion.zsh
```

2. Crie um novo diretório em seu diretório base para conter o script:

```
mkdir -p ~/.zsh/completions
```

3. Mova o script baixado para o diretório `~/.zsh/completions`:

```
mv tridentctl-completion.zsh ~/.zsh/completions/
```

4. Adicione a seguinte linha ao arquivo `~/.zprofile` no seu diretório base:

```
source ~/.zsh/completions/tridentctl-completion.zsh
```

Resultado

Na sua próxima sessão de login no shell, você pode usar o recurso de autocompletar comandos com o plugin tridentctl-protect.

Personalizar 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 instalar o Trident Protect. Definir limites de recursos permite controlar quanto dos recursos do cluster 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 arquivo de configuração mostra as configurações disponíveis e contém os valores padrão para cada limite de recurso:

```
---
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 arquivo `resourceLimits.yaml`:

```
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 (OpenShift SCCs) para os contêineres do Trident Protect após instalar o Trident Protect. Essas restrições definem limitações de segurança para 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 do Helm identificar um ambiente OpenShift. Se não estiver operando em 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. SCCs com valores de prioridade mais altos são avaliados antes daqueles com valores mais baixos.	1

3. Aplique os valores do arquivo `sccconfig.yaml`:

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect -f sccconfig.yaml --reuse-values
```

Isso substituirá os valores padrão pelos valores especificados no arquivo `sccconfig.yaml`.

Configurar configurações adicionais do helm chart do Trident Protect

Você pode personalizar as configurações do AutoSupport e o filtro de namespace para atender às suas necessidades específicas. A tabela a seguir descreve os parâmetros de configuração disponíveis:

Parâmetro	Tipo	Descrição
AutoSupport.proxy	string	Configura uma URL de proxy para conexões do NetApp AutoSupport. Use isto para rotear uploads de pacotes de suporte por meio de um servidor proxy. Exemplo: `http://my.proxy.url`.

Parâmetro	Tipo	Descrição
AutoSupport.insecure	booleano	Ignora a verificação TLS para conexões proxy do AutoSupport quando definido como `true`. Use somente para conexões proxy inseguras. (padrão: `false`)
AutoSupport.enabled	booleano	Ativa ou desativa o envio diário de pacotes do Trident Protect AutoSupport. Quando definido como `false`, os envios diários agendados são desativados, mas você ainda pode gerar pacotes de suporte manualmente. (padrão: `true`)
restoreSkipNamespaceAnnotations	string	Lista de anotações de namespace separadas por vírgulas para excluir das operações de backup e restauração. Permite filtrar namespaces com base em anotações.
restoreSkipNamespaceLabels	string	Lista de rótulos de namespace separados por vírgula para excluir das operações de backup e restauração. Permite filtrar namespaces com base nos rótulos.

Você pode configurar essas opções usando um arquivo de configuração YAML ou parâmetros de linha de comando:

Use o 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"
```

3. Após preencher o `values.yaml` file 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
```

Use a flag da CLI

Passos

1. Utilize o seguinte comando com a flag `--set` 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}" \
  --reuse-values
```

Restringir os pods do Trident Protect a nós específicos

Você pode usar a restrição de seleção de nós nodeSelector do Kubernetes para controlar quais dos seus nós estão qualificados para executar pods do Trident Protect, com base nos rótulos dos nós. Por padrão, 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 nodeSelector ao arquivo e modifique o arquivo para adicionar ou alterar rótulos de nós para restringir de acordo com as necessidades do seu ambiente. Por exemplo, o seguinte arquivo contém

a restrição padrão do sistema operacional, mas também direciona para uma região e um nome de aplicativo específicos:

```
nodeSelector:
  kubernetes.io/os: linux
  region: us-west
  app.kubernetes.io/name: mysql
```

3. Aplique os valores do arquivo `nodeSelectorConfig.yaml`:

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect -f nodeSelectorConfig.yaml --reuse-values
```

Isso substitui as restrições padrão pelas que você especificou no arquivo `nodeSelectorConfig.yaml`.

Gerenciar Trident Protect

Gerencie a autorização e o controle de acesso do Trident

Trident Protect utiliza o modelo Kubernetes de controle de acesso baseado em funções (RBAC). Por padrão, Trident Protect fornece um único namespace de sistema e sua respectiva conta de serviço padrão. Se você tem uma organização com muitos usuários ou necessidades de segurança específicas, pode usar os recursos de RBAC do Trident Protect para obter um controle mais granular sobre o acesso a recursos e namespaces.

O administrador do cluster sempre tem acesso aos recursos no namespace padrão `trident-protect` e também pode acessar recursos em todos os outros namespaces. Para controlar o acesso a recursos e aplicativos, você precisa criar namespaces adicionais e adicionar recursos e aplicativos a esses namespaces.

Observe que nenhum usuário pode criar CRs de gerenciamento de dados de aplicativos no namespace `trident-protect`padrão. Você precisa criar CRs de gerenciamento de dados de aplicativos em um namespace de aplicativo (como prática recomendada, crie CRs de gerenciamento de dados de aplicativos no mesmo namespace do aplicativo associado).

Somente os administradores devem ter acesso aos objetos de recursos personalizados privilegiados do Trident Protect, que incluem:

- **AppVault**: requer dados de credencial do bucket
- **AutoSupportBundle**: coleta métricas, registros e outros dados confidenciais do Trident Protect
- **AutoSupportBundleSchedule**: gerencia os agendamentos de coleta de logs

Como prática recomendada, use RBAC para restringir o acesso a objetos privilegiados a 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: gerenciar o acesso para dois grupos de usuários

Por exemplo, uma organização possui um administrador de cluster, um grupo de usuários de engenharia e um grupo de usuários de marketing. O administrador de cluster executaria as seguintes tarefas para criar um ambiente onde o grupo de engenharia e o grupo de marketing tenham acesso apenas aos recursos atribuídos aos seus respectivos namespaces.

Etapa 1: crie um espaço de nomes para conter os recursos de 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 os recursos em cada namespace

Cada novo namespace que você cria vem com uma conta de serviço padrão, mas você deve criar uma conta de serviço para cada grupo de usuários para que possa dividir ainda mais os privilégios 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
```

Etapa 3: crie um segredo para cada nova conta de serviço

Uma chave secreta da conta de serviço é usada para autenticar com a conta de serviço e pode ser facilmente excluída e recriada se for comprometida.

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
```

Etapa 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 Trident Protect. Você pode vincular esse 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 de 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
```

Etapa 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 os 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 de AppVault e segredo que conceda a um usuário acesso a um AppVault. Por exemplo, o seguinte CR 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 uma Role CR para permitir que os administradores do 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. Crie e aplique uma 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. Tentativa de recuperar informações do objeto AppVault para todos os namespaces:

   ```
   kubectl get appvaults -n trident-protect
   --as=system:serviceaccount:engineering-ns:eng-user
   ```

   Você deverá ver uma saída semelhante à 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 verificar se o usuário consegue obter as AppVault informações às quais agora tem permissão de acesso:

```
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 deverá ver uma saída semelhante à seguinte:

```
yes
```

Resultado

Os usuários aos quais você concedeu permissões do AppVault devem poder usar objetos AppVault autorizados para operações de gerenciamento de dados do aplicativo e não devem poder acessar recursos fora dos namespaces atribuídos nem criar novos recursos aos quais não tenham 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. Usá-lo com Trident Protect expõe informações úteis sobre o estado dos recursos em seu ambiente.

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. Juntos, kube-state-metrics e Prometheus fornecem uma maneira de monitorar a integridade e o status dos recursos que você gerencia com Trident Protect.

Alertmanager é um serviço que recebe os alertas enviados por ferramentas como Prometheus e os encaminha para destinos que você configurar.

> As configurações e orientações incluídas nestas etapas são apenas exemplos; você precisa personalizá-las para corresponder ao seu ambiente. Consulte a seguinte documentação oficial para instruções específicas e suporte:
>
> - "documentação do kube-state-metrics"
> - "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 kube-state-metrics, Prometheus e Alertmanager.

Instalar kube-state-metrics

Você pode instalar kube-state-metrics usando Helm.

Passos

1. Adicione o gráfico Helm kube-state-metrics. Por exemplo:

```
helm repo add prometheus-community https://prometheus-
community.github.io/helm-charts
helm repo update
```

2. Aplique o CRD ServiceMonitor do Prometheus 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 Helm chart (por exemplo, `metrics-config.yaml`). Você pode personalizar a seguinte configuração de exemplo para corresponder ao seu ambiente:

metrics-config.yaml: configuração do Helm chart 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 o kube-state-metrics 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 do recurso personalizado kube-state-metrics".

Instale o 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".

Etapa 2: configure as ferramentas de monitoramento para funcionarem em conjunto

Após instalar as ferramentas de monitoramento, você precisa configurá-las para que funcionem juntas.

Passos

1. Integre o kube-state-metrics com o Prometheus. Edite o arquivo de configuração do Prometheus (`prometheus.yaml` e adicione as informações do serviço kube-state-metrics. Por exemplo:

   prometheus.yaml: integração do serviço kube-state-metrics com 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 o Prometheus para encaminhar alertas para o Alertmanager. Edite o arquivo de configuração do 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 do kube-state-metrics e enviar alertas para Alertmanager. Agora você está pronto para configurar quais condições acionam um alerta e para onde os alertas devem ser enviados.

Etapa 3: configurar alertas e destinos de alerta

Depois de configurar as ferramentas para funcionarem em conjunto, você precisa configurar que tipo de informação aciona alertas e para onde os alertas devem ser enviados.

Exemplo de alerta: falha no 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 este trecho de YAML em seu arquivo de configuração `prometheus.yaml`:

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 respectiva configuração no arquivo `alertmanager.yaml`.

O exemplo a seguir configura o Alertmanager para enviar notificações a um canal do Slack. Para personalizar este exemplo para o seu ambiente, substitua o valor da `api_url` chave pela URL do webhook do Slack usada em seu 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
```

Gerar um pacote de suporte Trident Protect

Trident Protect permite que os administradores gerem pacotes que incluem informações úteis para o NetApp Support, incluindo logs, métricas e informações de topologia sobre os clusters e aplicativos sob gerenciamento. Se você estiver conectado à internet, poderá enviar pacotes de suporte para o site de suporte da NetApp (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 dê um nome a ele (por exemplo, `trident-protect-support-bundle.yaml`).
2. Configurar os seguintes atributos:
   - **metadata.name**: (*Obrigatório*) O nome deste recurso personalizado; escolha um nome único e adequado ao seu ambiente.
   - **spec.triggerType**: (*Obrigatório*) Determina se o pacote de suporte é gerado imediatamente ou agendado. A geração agendada do pacote ocorre às 00h UTC. Valores possíveis:
     - Agendado
     - Manual
   - **spec.uploadEnabled**: (*Opcional*) Controla se o pacote de suporte deve ser carregado no site de suporte da NetApp após ser gerado. Se não for especificado, o padrão é `false`. Valores possíveis:
     - verdadeiro
     - falso (default)
   - **spec.dataWindowStart**: (*Opcional*) Uma string de data no formato RFC 3339 que especifica a data e hora em que a janela de dados incluída no pacote de suporte deve começar. Se não for especificado, o padrão é 24 horas atrás. A data mais antiga da janela que você pode especificar é 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. Após preencher o arquivo `trident-protect-support-bundle.yaml` com os valores corretos, aplique a 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 os valores entre colchetes pelas informações do seu ambiente.

O `trigger-type` determina se o pacote será criado imediatamente ou se o horário de criação será definido pelo agendamento, e pode ser `Manual` ou `Scheduled`. A configuração padrão 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

Após criar um pacote de suporte usando qualquer um dos métodos, você pode monitorar o progresso da geração e recuperá-lo para seu sistema local.

Passos

1. Aguarde até que o `status.generationState` atinja o estado `Completed`. 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 o seu sistema local. Obtenha o comando de cópia do pacote AutoSupport concluído:

   ```
   kubectl describe autosupportbundle trident-protect-support-bundle -n trident-protect
   ```

   Localize o comando `kubectl cp` na saída e execute-o, substituindo o argumento de destino pelo diretório local de sua preferência.

Atualizar Trident Protect

Você pode atualizar Trident Protect para a versão mais recente para aproveitar novos recursos ou correções de bugs.

- Ao atualizar da versão 24.10, snapshots em execução durante a atualização podem falhar. Essa falha não impede que snapshots futuros, sejam manuais ou agendados, sejam criados. Se um snapshot falhar durante a atualização, você pode criar manualmente um novo snapshot para garantir que seu aplicativo esteja protegido.

  Para evitar possíveis falhas, você pode desativar todos os agendamentos de snapshots antes da atualização e reativá-los posteriormente. No entanto, isso resulta na perda de quaisquer snapshots agendados durante o período de atualização.

- Para instalações em registros privados, certifique-se de que o Helm chart e as imagens necessárias para a versão de destino estejam disponíveis em seu registro privado e verifique se seus valores Helm personalizados são compatíveis com a nova versão do chart. Para obter mais informações, consulte "Instale 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 versão 26.02:
  a. Primeira atualização da 24.10 para a 25.02.
  b. Em seguida, atualize de 25.02 para 26.02.

Etapa 2: atualize Trident Protect

Para atualizar Trident Protect, execute as seguintes etapas.

Passos

1. Atualize o repositório do Trident:

```
helm repo update
```

2. Atualize os CRDs do Trident Protect:

   Esta etapa é necessária se você estiver atualizando de uma versão anterior à 25.06, pois os CRDs agora estão incluídos no Helm chart do Trident Protect.

   a. Execute este comando para transferir 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` chart:

> Não desinstale o `trident-protect-crds` chart 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. Atualizar Trident Protect:

```
helm upgrade trident-protect netapp-trident-protect/trident-protect
--version 100.2602.0 --namespace trident-protect
```

> 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 solução de problemas, pois ajuda a equipe de 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 Trident Protect é conhecido como um AppVault. Os objetos AppVault são a representação declarativa do fluxo de trabalho do Kubernetes de um bucket de armazenamento. Um CR AppVault contém as configurações necessárias para que um bucket seja usado em operações de proteção, como backups, snapshots, operações de restauração e replicação SnapMirror. Somente administradores podem criar AppVaults.

Você precisa criar um AppVault CR manualmente ou pela linha de comando ao executar operações de proteção de dados em um aplicativo. O AppVault CR é específico para o seu ambiente, e você pode usar os exemplos desta página como guia ao criar AppVault CRs.

> Certifique-se de que o AppVault CR esteja no cluster onde Trident Protect está instalado. Se o AppVault CR não existir ou você não conseguir acessá-lo, a linha de comando exibirá um erro.

Configurar autenticação e senhas do AppVault

Antes de criar um AppVault CR, certifique-se de que o AppVault e o data mover escolhido possam se autenticar com o provedor e quaisquer recursos relacionados.

Senhas do repositório do Data Mover

Ao criar objetos AppVault usando CRs ou o plugin CLI do Trident Protect, você pode especificar um segredo do Kubernetes com senhas personalizadas para criptografia Restic e Kopia. Se você não especificar um segredo, Trident Protect usará uma senha padrão.

- Ao criar manualmente CRs do AppVault, use o campo **spec.dataMoverPasswordSecretRef** para especificar o segredo.
- Ao criar objetos AppVault usando a Trident Protect CLI, utilize o argumento `--data-mover-password-secret-ref` para especificar o segredo.

Crie um segredo de senha do repositório de Data Mover

Use os exemplos a seguir para criar a senha secreta. Ao criar AppVault objetos, você pode instruir Trident Protect a usar essa senha secreta para autenticar com o repositório do data mover.

- Dependendo do gerenciador de dados que você estiver usando, você só precisa incluir a senha correspondente para esse gerenciador de dados. Por exemplo, se você estiver usando Restic e não planeja usar Kopia no futuro, você pode incluir apenas a senha do Restic ao 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 o 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 IAM de armazenamento compatível com S3

Ao acessar armazenamento compatível com S3, como Amazon S3, S3 genérico, "StorageGrid S3", ou "ONTAP S3" usando Trident Protect, você precisa garantir que as credenciais do usuário fornecidas tenham as permissões necessárias para acessar o bucket. O exemplo a seguir mostra uma política que concede as permissões mínimas necessárias para acesso com Trident Protect. Você pode aplicar essa política ao usuário que gerencia as políticas de buckets compatíveis com 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 em "Documentação do Amazon S3".

Identidade do Pod EKS para autenticação Amazon S3

Trident Protect oferece suporte à EKS Pod Identity para operações de movimentação de dados do Kopia. Esse recurso permite acesso seguro a buckets do S3 sem armazenar credenciais da AWS em segredos do Kubernetes.

Requisitos para EKS Pod Identity com Trident Protect

Antes de usar o EKS Pod Identity com Trident Protect, certifique-se do seguinte:

- Seu cluster EKS tem Pod Identity habilitado.
- Você criou uma função do IAM com as permissões necessárias para o bucket do S3. Para saber mais, consulte "Permissões 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 do AWS EKS Pod Identity".

AppVault Configuração Ao usar o EKS Pod Identity, configure seu AppVault CR com a `useIAM: true` flag 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
```

AppVault exemplos de geração de chaves 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 autenticação IAM. A forma como você gera as chaves para as credenciais varia de acordo com o provedor. A seguir estão exemplos de geração de chaves pela linha de comando para vários provedores. Você pode usar os seguintes exemplos 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
```

S3 genérico

```
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
```

AppVault creation exemplos

A seguir estão exemplos de definições do AppVault para cada provedor.

AppVault CR exemplos

Você pode usar os seguintes exemplos de CR para criar objetos AppVault para cada provedor de nuvem.

- Opcionalmente, você pode especificar um segredo do Kubernetes que contenha senhas personalizadas para a criptografia do repositório Restic e Kopia. Consulte Senhas do repositório do Data Mover para mais informações.
- Para objetos AppVault do Amazon S3 (AWS), você pode opcionalmente especificar um sessionToken, o que é útil se você estiver usando logon único (SSO) para autenticação. Esse token é criado quando você gera chaves para o provedor em AppVault exemplos de geração de chaves para provedores de nuvem.
- Para objetos S3 AppVault, você pode opcionalmente especificar uma URL de proxy de saída para o tráfego S3 de saída usando a chave `spec.providerConfig.S3.proxyURL`.

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
```

Para ambientes EKS usando Pod Identity com Kopia data mover, você pode remover a `providerCredentials` seção e adicionar `useIAM: true` sob a configuração `s3` 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
```

S3 genérico

```
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
```

AppVault exemplos de criação usando o Trident Protect CLI

Você pode usar os seguintes exemplos de comandos da CLI para criar AppVault CRs para cada provedor.

- Opcionalmente, você pode especificar um segredo do Kubernetes que contenha senhas personalizadas para a criptografia do repositório Restic e Kopia. Consulte Senhas do repositório do Data Mover para mais informações.
- Para objetos S3 AppVault, você pode opcionalmente especificar uma URL de proxy de saída para o 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
```

S3 genérico

```
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
```

Opções de `providerConfig.s3` configuração suportadas

Consulte a tabela a seguir para as opções de configuração do provedor S3:

Parâmetro	Descrição	Padrão	Exemplo
`providerConfig.s3.skipCertValidation`	Desative a verificação de certificado SSL/TLS.	falso	"true", "false"
`providerConfig.s3.secure`	Habilite a comunicação HTTPS segura com o endpoint S3.	verdadeiro	"true", "false"
`providerConfig.s3.proxyURL`	Especifique a URL do servidor proxy usada 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 buckets do S3. Aplicável para EKS Pod Identity.	falso	true, false

Ver informações do AppVault

Você pode usar o plugin Trident Protect CLI para visualizar informações sobre objetos AppVault que você criou no cluster.

Passos

1. Visualize 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 visualizar o AppVaultPath de cada recurso, use a flag `--show-paths`.

   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 via helm. Por exemplo: `--set clusterName=production1`.

Remover um AppVault

Você pode remover um objeto AppVault a qualquer momento.

Não remova a chave `finalizers` no CR AppVault antes de excluir o objeto AppVault. Se você fizer isso, pode resultar em dados residuais no bucket AppVault e recursos órfãos no cluster.

Antes de começar

Certifique-se de ter excluído todos os CRs de snapshot e backup que estão sendo usados pelo AppVault que você deseja excluir.

Remova um AppVault usando a CLI do Kubernetes

1. Remova o objeto AppVault, substituindo `appvault-name` pelo nome do objeto AppVault a ser removido:

```
kubectl delete appvault <appvault-name> \
-n trident-protect
```

Remova um AppVault usando o Trident Protect CLI

1. Remova o objeto AppVault, substituindo `appvault-name` pelo nome do objeto AppVault a ser removido:

```
tridentctl-protect delete appvault <appvault-name> \
-n trident-protect
```

Defina uma aplicação para gestão com Trident Protect

Você pode definir um aplicativo que deseja gerenciar com Trident Protect criando um CR de aplicativo e um CR de AppVault associado.

Criar 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 Trident Protect está instalado. O AppVault CR é específico para o seu ambiente; para exemplos de AppVault CRs, consulte "Recursos personalizados do AppVault."

Definir uma aplicação

Você precisa definir cada aplicativo que deseja gerenciar com Trident Protect. Você pode definir um aplicativo para gerenciamento criando manualmente um CR do aplicativo ou usando a CLI do Trident Protect.

Adicionar um aplicativo usando um CR

Passos

1. Crie o arquivo CR do aplicativo de destino:
   a. Crie o arquivo de recurso personalizado (CR) e dê um nome a ele (por exemplo, `maria-app.yaml`).
   b. Configurar os seguintes atributos:
      - **metadata.name**: (*Obrigatório*) O nome do recurso personalizado do aplicativo. Anote o nome escolhido, pois outros arquivos CR necessários para operações de proteção fazem referência a esse valor.
      - **spec.includedNamespaces**: (*Obrigatório*) Use o seletor de namespace e rótulo para especificar os namespaces e recursos que o aplicativo utiliza. O namespace do aplicativo deve fazer parte desta lista. O seletor de rótulo é opcional e pode ser usado para filtrar recursos dentro de cada namespace especificado.
      - **spec.includedClusterScopedResources**: (*Opcional*) Use este atributo para especificar recursos com escopo de cluster a serem incluídos na definição do aplicativo. Este atributo permite selecionar esses recursos com base em seu grupo, versão, tipo e rótulos.
        - **groupVersionKind**: (*Obrigatório*) Especifica o grupo de API, a versão e o tipo do recurso com escopo de cluster.
        - **labelSelector**: (Opcional) Filtra os recursos com escopo de cluster com base em seus rótulos.
      - **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 KubeVirt ambientes, onde o congelamento do sistema de arquivos ocorre antes dos snapshots. Especifique se esta aplicação pode gravar no sistema de arquivos durante um snapshot. Se definida como true, a aplicação ignora a configuração global e pode gravar no sistema de arquivos durante um snapshot. Se definida como false, a aplicação ignora a configuração global e o sistema de arquivos é congelado durante um snapshot. Se especificada, mas a aplicação não tiver máquinas virtuais na definição da aplicação, a anotação será ignorada. Se não especificada, a aplicação segue a "configuração global de congelamento do Trident Protect".

Se você precisar aplicar essa anotação depois que um aplicativo já tiver sido criado, 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**: (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 no "Documentação do Kubernetes". Por exemplo: `"trident.netapp.io/os=linux"`.

> Quando ambos `resourceFilter` e `labelSelector` são usados, `resourceFilter` é executado primeiro e, em seguida, `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. Após criar a CR do aplicativo para corresponder ao seu ambiente, aplique a 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 os valores entre colchetes pelas informações do seu 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 se aplica somente a aplicativos definidos a partir de máquinas virtuais, como em KubeVirt ambientes, onde o congelamento do sistema de arquivos ocorre antes dos snapshots. Se você definir a anotação como `true`, o aplicativo ignora a configuração global e pode gravar no sistema de arquivos durante um snapshot. Se você definir como `false`, o aplicativo ignora a configuração global e o sistema de arquivos é congelado durante 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 a "configuração global de congelamento 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 a definição local 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":"Persis
tentVolume","Version":"v1"},{"Kind":"VolumeSnapshotClass","Versio
n":"v1"}]' -n <my-app-namespace>
```

Proteja aplicativos usando Trident Protect

Você pode proteger todos os aplicativos gerenciados pelo Trident Protect tirando snapshots e backups usando uma política de proteção automatizada ou sob demanda.

Você pode configurar 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 Trident Protect".

Criar um snapshot sob demanda

Você pode criar um snapshot sob demanda a qualquer momento.

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 que você criou, configure os seguintes atributos:
   - **metadata.name**: (*Obrigatório*) O nome deste recurso personalizado; escolha um nome único e adequado ao seu ambiente.
   - **spec.applicationRef**: o nome do aplicativo no Kubernetes para o qual será criado o Snapshot.
   - **spec.appVaultRef**: (*Obrigatório*) O nome do AppVault onde o conteúdo do Snapshot (metadados) deve ser armazenado.
   - **spec.reclaimPolicy**: (*Opcional*) Define o que acontece com o AppArchive de um snapshot quando o CR do snapshot é excluído. Isso significa que mesmo quando definido como `Retain`, o snapshot será excluído. Opções válidas:
     - `Retain` (padrão)
     - `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. Após preencher o arquivo `trident-protect-snapshot-cr.yaml` com os valores corretos, aplique a CR:

   ```
   kubectl apply -f trident-protect-snapshot-cr.yaml
   ```

Criar um instantâneo usando a CLI

Passos

1. Crie o instantâneo, substituindo os valores entre colchetes pelas 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 app a qualquer momento.

> 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 o tempo de expiração do token de sessão da AWS seja suficiente para quaisquer operações de backup do s3 de longa duração. Se o token expirar durante a operação de backup, a operação pode falhar.

- Consulte o "Documentação do AWS API" para mais informações sobre como verificar a expiração do token de sessão atual.
- Consulte "Documentação do AWS IAM" para obter mais informações sobre credenciais com recursos da AWS.

Criar um backup usando um CR

Passos

1. Crie o arquivo de recurso personalizado (CR) e nomeie-o `trident-protect-backup-cr.yaml`.
2. No arquivo que você criou, configure os seguintes atributos:
   - **metadata.name**: (*Obrigatório*) O nome deste recurso personalizado; escolha um nome único e adequado ao seu ambiente.
   - **spec.applicationRef**: (*Obrigatório*) O nome do aplicativo Kubernetes a ser feito backup.
   - **spec.appVaultRef**: (*Obrigatório*) O nome do AppVault onde o conteúdo do backup deve ser armazenado.
   - **spec.dataMover**: (*Opcional*) Uma string indicando qual ferramenta de backup usar para a operação de backup. Valores possíveis (diferencia maiúsculas de minúsculas):
     - `Restic`
     - `Kopia` (padrão)
   - **spec.reclaimPolicy**: (*Opcional*) Define o que acontece com um backup quando liberado de sua reivindicação. Valores possíveis:
     - `Delete`
     - `Retain` (padrão)
   - **spec.snapshotRef**: (*Opcional*): Nome do snapshot a ser usado como origem do backup. Se não for fornecido, um snapshot temporário será criado e feito backup.

     Exemplo YAML:

```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. Após preencher o arquivo `trident-protect-backup-cr.yaml` com os valores corretos, aplique a CR:

```
kubectl apply -f trident-protect-backup-cr.yaml
```

Criar um backup usando a CLI

Passos

1. Crie o backup, substituindo os valores entre colchetes pelas 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>
```

Você pode opcionalmente usar a `--full-backup` flag para especificar se um backup deve ser não incremental. Por padrão, todos os backups são incrementais. Quando essa flag é usada, o backup se torna não incremental. A melhor prática é realizar um backup completo periodicamente e depois realizar backups incrementais entre os backups completos para minimizar o risco associado a restaurações.

Anotações de backup suportadas

A tabela a seguir descreve as anotações que você pode usar ao criar um backup CR:

Anotação	Tipo	Descrição	Valor padrão
protect.trident.netapp.io/backup-completo	string	Especifica se um backup deve ser não incremental. Defina como `true` para criar um backup não incremental. A melhor prática é realizar um backup completo periodicamente e depois realizar backups incrementais entre os backups completos para minimizar o risco associado a restaurações.	"false"
protect.trident.netapp.io/snapshot-completion-timeout	string	O tempo máximo permitido para a conclusão geral da operação de Snapshot.	"60m"
protect.trident.netapp.io/volume-snapshots-ready-to-use-timeout	string	Tempo máximo permitido para que os snapshots de volume atinjam o estado pronto para uso.	"30m"
protect.trident.netapp.io/volume-snapshots-created-timeout	string	O tempo máximo permitido para que snapshots de volume sejam criados.	"5m"
protect.trident.netapp.io/pvc-bind-timeout-sec	string	Tempo máximo (em segundos) de espera para que qualquer PersistentVolumeClaims (PVC) recém-criado atinja a `Bound` fase antes que a operação falhe.	"1200" (20 minutos)

Crie um cronograma de proteção de dados

Uma política de proteção protege um app criando snapshots, backups ou ambos em uma programação definida. Você pode optar por criar snapshots e backups a cada hora, dia, semana e mês, e pode especificar o número de cópias a reter. Você pode agendar um backup completo não incremental usando a anotação full-backup-rule. Por padrão, todos os backups são incrementais. Realizar um backup completo periodicamente, juntamente com backups incrementais entre eles, ajuda a reduzir o risco associado a restaurações.

- Você pode criar agendamentos para snapshots apenas definindo `backupRetention` como zero e `snapshotRetention` como um valor maior que zero. Definir `snapshotRetention` como zero significa que qualquer backup agendado ainda criará snapshots, mas estes são temporários e são 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 um cronograma usando uma CR

Passos

1. Crie o arquivo de recurso personalizado (CR) e nomeie-o `trident-protect-schedule-cr.yaml`.
2. No arquivo que você criou, configure os seguintes atributos:
   - **metadata.name**: (*Obrigatório*) O nome deste recurso personalizado; escolha um nome único e adequado ao seu ambiente.
   - **spec.dataMover**: (*Opcional*) Uma string indicando qual ferramenta de backup usar para a operação de backup. Valores possíveis (diferencia maiúsculas de minúsculas):
     - `Restic`
     - `Kopia` (padrão)
   - **spec.applicationRef**: o nome do aplicativo no Kubernetes para o qual será feito o backup.
   - **spec.appVaultRef**: (*Obrigatório*) O nome do AppVault onde o conteúdo do backup deve ser armazenado.
   - **spec.backupRetention**: (*Obrigatório*) O número de backups a serem mantidos. Zero indica que nenhum backup deve ser criado (apenas snapshots).
   - **spec.backupReclaimPolicy**: (*Opcional*) Determina o que acontece com um backup se o backup CR for excluído durante seu período de retenção. Após o período de retenção, os backups são sempre excluídos. Valores possíveis (diferencia maiúsculas de minúsculas):
     - `Retain` (padrão)
     - `Delete`
   - **spec.snapshotRetention**: (*Obrigatório*) O número de snapshots a serem mantidos. Zero indica que nenhum snapshot 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 snapshots são sempre excluídos. Valores possíveis (diferencia maiúsculas de minúsculas):
     - `Retain`
     - `Delete` (padrão)
   - **spec.granularity**: A frequência com que o agendamento deve ser executado. Valores possíveis, juntamente com os 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 a programação deve ser executada. 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 backup e agendamento de 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 agendamento somente com 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. Após preencher o arquivo `trident-protect-schedule-cr.yaml` com os valores corretos, aplique a CR:

```
kubectl apply -f trident-protect-schedule-cr.yaml
```

Crie um agendamento usando a linha de comando (CLI)

Passos

1. Crie o cronograma de proteção, substituindo os valores entre colchetes pelas informações do seu ambiente. Por exemplo:

Você pode usar `tridentctl-protect create schedule --help` para visualizar informações de ajuda detalhadas 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 de backup completo**: Use a `--full-backup-rule` flag para agendar backups completos não incrementais. Esta flag funciona apenas 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.

    A `--full-backup-rule`opção 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 de instantâneo**: Defina `--backup-retention 0` e especifique um valor maior que zero para `--snapshot-retention`.

Anotações de cronograma suportadas

A tabela a seguir descreve as anotações que você pode usar ao criar um schedule CR:

Anotação	Tipo	Descrição	Valor padrão
protect.trident.netapp.io/full-backup-rule	string	Especifica a regra para agendamento de backups completos. Você pode configurá-la para `Always` backup completo ou personalizá-la de acordo com suas necessidades. Por exemplo, se você escolher granularidade diária, poderá especificar os dias da semana em que o backup completo deve ocorrer (por exemplo, `"Monday,Thursday"`). Os valores válidos para os dias da semana são: Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday. Observe que esta anotação só pode ser usada com agendamentos que tenham `granularity` definido como `Daily`.	Não configurado (todos os backups são incrementais)
protect.trident.netapp.io/snapshot-completion-timeout	string	O tempo máximo permitido para a conclusão geral da operação de Snapshot.	"60m"
protect.trident.netapp.io/volume-snapshots-ready-to-use-timeout	string	Tempo máximo permitido para que os snapshots de volume atinjam o estado pronto para uso.	"30m"
protect.trident.netapp.io/volume-snapshots-created-timeout	string	O tempo máximo permitido para que snapshots de volume sejam criados.	"5m"
protect.trident.netapp.io/pvc-bind-timeout-sec	string	Tempo máximo (em segundos) de espera para que qualquer PersistentVolumeClaims (PVC) recém-criado atinja a `Bound` fase antes que a operação falhe.	"1200" (20 minutos)

Excluir um instantâneo

Exclua os snapshots agendados ou sob demanda que você não precisa mais.

Passos

1. Remova o snapshot CR associado ao snapshot:

```
kubectl delete snapshot <snapshot_name> -n my-app-namespace
```

Excluir um backup

Exclua os backups agendados ou sob demanda que você não precisa mais.

Certifique-se de que a política de recuperação esteja configurada para `Delete` remover todos os dados de backup do storage de objetos. A configuração padrão da política é `Retain` para evitar perda de dados. Se a política não for alterada para `Delete`, os dados de backup permanecerão no storage de objetos e precisarão ser excluídos manualmente.

Passos

1. Remova o backup CR 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 que está em andamento, foi concluída ou falhou.

Passos

1. Utilize o seguinte comando para recuperar o status da operação de backup, substituindo os valores entre colchetes pelas informações do seu ambiente:

```
kubectl get backup -n <namespace_name> <my_backup_cr_name> -o jsonpath
='{.status}'
```

Habilite backup e restauração para operações do azure-netapp-files (ANF)

Se você instalou 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 funciona com volumes NFSv4 e não consome espaço adicional do pool de capacidade.

Antes de começar

Certifique-se do seguinte:

- Você instalou Trident Protect.
- Você definiu um aplicativo no Trident Protect. Este aplicativo terá funcionalidade de proteção limitada até que você conclua este procedimento.
- Você selecionou `azure-netapp-files` como classe de armazenamento padrão para seu backend de armazenamento.

Expandir para ver as etapas de configuração

1. Execute o seguinte no Trident se o volume ANF foi criado antes da atualização para Trident 24.10:

   a. Habilite o diretório de instantâneos para cada PV baseado em azure-netapp-files e associado à aplicação:

   ```
   tridentctl update volume <pv name> --snapshot-dir=true -n trident
   ```

   b. Confirme se o diretório de snapshot foi habilitado para cada PV associado:

   ```
   tridentctl get volume <pv name> -n trident -o yaml | grep
   snapshotDir
   ```

   Resposta:

   ```
   snapshotDirectory: "true"
   ```

   +
   Quando o diretório de snapshots não está habilitado, 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 feito backup.

Resultado

O aplicativo está pronto para backup e restauração usando Trident Protect. Cada PVC também está disponível para ser usado por outros aplicativos para backup e restauração.

Restaurar aplicativos

Restaure aplicativos usando Trident Protect

Você pode usar Trident Protect para restaurar seu aplicativo a partir de um snapshot ou backup. Restaurar a partir de um snapshot existente será mais rápido ao restaurar o aplicativo para o mesmo cluster.

- Ao restaurar um aplicativo, todos os ganchos de execução configurados para o aplicativo são restaurados juntamente com o aplicativo. Se houver um gancho de execução pós-restauração, ele é executado automaticamente como parte da operação de restauração.
- A restauração a partir de um backup para um namespace diferente ou para o namespace original é suportada para volumes qtree. No entanto, a restauração a partir 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 "Use as configurações avançadas de restauração do Trident Protect".

Restaurar a partir de um backup para um namespace diferente

Ao restaurar um backup para um namespace diferente usando um BackupRestore CR, Trident Protect restaura o aplicativo em um novo namespace e cria um CR de aplicativo para o aplicativo restaurado. Para proteger o aplicativo restaurado, crie backups ou snapshots sob demanda, ou estabeleça um agendamento de proteção.

- 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 do backup, exclua e recrie o namespace de destino ou restaure o backup para um novo namespace.
- Ao usar uma CR para restaurar para um novo namespace, você deve criar manualmente o namespace de destino antes de aplicar a CR. Trident Protect cria namespaces automaticamente apenas quando se usa a CLI.

Antes de começar

Certifique-se de que o tempo de expiração do token de sessão da AWS seja suficiente para quaisquer operações de restauração do s3 de longa duração. Se o token expirar durante a operação de restauração, a operação pode falhar.

- Consulte o "Documentação do AWS API" para mais informações sobre como verificar a expiração do token de sessão atual.
- Consulte "Documentação do AWS IAM" para obter mais informações sobre credenciais com recursos da AWS.

Ao restaurar backups usando Kopia como o data mover, você pode opcionalmente especificar anotações no CR ou usando a CLI para controlar o comportamento do armazenamento temporário usado pelo Kopia. Consulte o "Documentação Kopia" para mais informações sobre as opções que você pode configurar. Use o comando `tridentctl-protect create --help` para mais informações sobre como especificar anotações com a Trident Protect CLI.

Use um CR

Passos

1. Crie o arquivo de recurso personalizado (CR) e nomeie-o `trident-protect-backup-restore-cr.yaml`.
2. No arquivo que você criou, configure os seguintes atributos:
   - **metadata.name**: (*Obrigatório*) O nome deste recurso personalizado; escolha um nome único e adequado ao seu ambiente.
   - **spec.appArchivePath**: O caminho dentro do AppVault onde o conteúdo do backup está armazenado. Você pode usar o seguinte comando para encontrar esse caminho:

     ```
     kubectl get backups <BACKUP_NAME> -n my-app-namespace -o
     jsonpath='{.status.appArchivePath}'
     ```

   - **spec.appVaultRef**: (*Obrigatório*) O nome do AppVault onde o conteúdo do backup está 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` pelas 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 precisar selecionar apenas determinados recursos do aplicativo para restaurar, adicione filtros que incluam ou excluam recursos marcados com rótulos específicos:

   > 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 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. Após preencher o arquivo `trident-protect-backup-restore-cr.yaml` com os valores corretos, aplique a CR:

```
kubectl apply -f trident-protect-backup-restore-cr.yaml
```

Use o CLI

Passos

1. Restaure o backup para um namespace diferente, substituindo os valores entre colchetes pelas informações do seu ambiente. O `namespace-mapping` argumento usa namespaces separados por

dois pontos para mapear os 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>
```

Restaurar 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 o tempo de expiração do token de sessão da AWS seja suficiente para quaisquer operações de restauração do s3 de longa duração. Se o token expirar durante a operação de restauração, a operação pode falhar.

- Consulte o "Documentação do AWS API" para mais informações sobre como verificar a expiração do token de sessão atual.
- Consulte "Documentação do AWS IAM" para obter mais informações sobre credenciais com recursos da AWS.

> ⓘ Ao restaurar backups usando Kopia como o data mover, você pode opcionalmente especificar anotações no CR ou usando a CLI para controlar o comportamento do armazenamento temporário usado pelo Kopia. Consulte o "Documentação Kopia" para mais informações sobre as opções que você pode configurar. Use o comando `tridentctl-protect create --help` para mais informações sobre como especificar anotações com a Trident Protect CLI.

Use um CR

Passos

1. Crie o arquivo de recurso personalizado (CR) e nomeie-o `trident-protect-backup-ipr-cr.yaml`.
2. No arquivo que você criou, configure os seguintes atributos:
   - **metadata.name**: (*Obrigatório*) O nome deste recurso personalizado; escolha um nome único e adequado ao seu ambiente.
   - **spec.appArchivePath**: O caminho dentro do AppVault onde o conteúdo do backup está armazenado. Você pode usar o seguinte comando para encontrar esse caminho:

     ```
     kubectl get backups <BACKUP_NAME> -n my-app-namespace -o jsonpath='{.status.appArchivePath}'
     ```

   - **spec.appVaultRef**: (*Obrigatório*) O nome do AppVault onde o conteúdo do backup está 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 precisar selecionar apenas determinados recursos do aplicativo para restaurar, adicione filtros que incluam ou excluam recursos marcados com rótulos específicos:

   > 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 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. Após preencher o arquivo `trident-protect-backup-ipr-cr.yaml` com os valores corretos, aplique a CR:

```
kubectl apply -f trident-protect-backup-ipr-cr.yaml
```

Use o CLI

Passos

1. Restaure o backup para o namespace original, substituindo os valores entre colchetes pelas 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>
```

Restaurar de um backup para um cluster diferente

Você pode restaurar um backup em um cluster diferente se houver um problema com o cluster original.

- Ao restaurar backups usando Kopia como o data mover, você pode opcionalmente especificar anotações no CR ou usando a CLI para controlar o comportamento do armazenamento temporário usado pelo Kopia. Consulte o "Documentação Kopia" para mais informações sobre as opções que você pode configurar. Use o comando `tridentctl-protect create --help` para mais informações sobre como especificar anotações com a Trident Protect CLI.
- Ao usar uma CR para restaurar para um novo namespace, você deve criar manualmente o namespace de destino antes de aplicar a CR. Trident Protect cria namespaces automaticamente apenas quando se usa a CLI.

Antes de começar

Certifique-se de que os seguintes pré-requisitos sejam atendidos:

- O cluster de destino tem Trident Protect instalado.
- O cluster de destino tem acesso ao caminho do bucket do mesmo AppVault que o cluster de origem, onde o backup está armazenado.
- Certifique-se de que seu ambiente local possa se conectar ao bucket de storage de objetos definido no AppVault CR ao executar o comando `tridentctl-protect get appvaultcontent`. Se as restrições de rede impedirem o acesso, execute o Trident Protect CLI a partir de um pod no cluster de destino.
- Certifique-se de que o tempo de 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 o "Documentação do AWS API" para mais informações sobre como verificar a expiração do token de sessão atual.
  - Consulte "Documentação da 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>
```

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 este comando exibe os backups disponíveis no AppVault, incluindo seus 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 no cluster de destino usando o nome AppVault e o caminho do arquivo:

> 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 que você criou, configure os seguintes atributos:
   - **metadata.name**: (*Obrigatório*) O nome deste recurso personalizado; escolha um nome único e adequado ao seu ambiente.
   - **spec.appVaultRef**: (*Obrigatório*) O nome do AppVault onde o conteúdo do backup está 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` pelas 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. Após preencher o arquivo `trident-protect-backup-restore-cr.yaml` com os valores corretos, aplique a CR:

```
kubectl apply -f trident-protect-backup-restore-cr.yaml
```

Use o CLI

1. Use o seguinte comando para restaurar o aplicativo, substituindo os valores entre colchetes pelas informações do seu ambiente. O argumento namespace-mapping usa namespaces separados por dois pontos para mapear os 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>
```

Restaurar a partir 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 SnapshotRestore CR, Trident Protect restaura o aplicativo em um novo namespace e cria um CR de aplicativo para o aplicativo restaurado. Para proteger o aplicativo restaurado, crie backups ou snapshots sob demanda, ou estabeleça um agendamento de proteção.

- SnapshotRestore é compatível com o atributo `spec.storageClassMapping`, mas somente quando as classes de armazenamento de origem e destino usam o mesmo backend de armazenamento. Se você tentar restaurar para uma `StorageClass` que usa um backend de armazenamento diferente, a operação de restauração falhará.
- Ao usar uma CR para restaurar para um novo namespace, você deve criar manualmente o namespace de destino antes de aplicar a CR. Trident Protect cria namespaces automaticamente apenas quando se usa a CLI.

Antes de começar

Certifique-se de que o tempo de expiração do token de sessão da AWS seja suficiente para quaisquer operações de restauração do s3 de longa duração. Se o token expirar durante a operação de restauração, a operação pode falhar.

- Consulte o "Documentação do AWS API" para mais informações sobre como verificar a expiração do token de sessão atual.
- Consulte "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 que você criou, configure os seguintes atributos:
   - **metadata.name**: (*Obrigatório*) O nome deste recurso personalizado; escolha um nome único e adequado ao seu ambiente.
   - **spec.appVaultRef**: (*Obrigatório*) O nome do AppVault onde o conteúdo do snapshot está armazenado.
   - **spec.appArchivePath**: O caminho dentro do AppVault onde o conteúdo do snapshot está armazenado. Você pode usar o seguinte comando para encontrar esse 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` pelas informações do seu ambiente.

     ```
     ---
     apiVersion: protect.trident.netapp.io/v1
     kind: SnapshotRestore
     metadata:
       name: my-cr-name
       namespace: my-app-namespace
     spec:
       appVaultRef: appvault-name
       appArchivePath: my-snapshot-path
       namespaceMapping: [{"source": "my-source-namespace",
     "destination": "my-destination-namespace"}]
     ```

3. (*Opcional*) Se precisar selecionar apenas determinados recursos do aplicativo para restaurar, adicione filtros que incluam ou excluam recursos marcados com rótulos específicos:

   > 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 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. Após preencher o arquivo `trident-protect-snapshot-restore-cr.yaml` com os valores corretos, aplique a CR:

```
kubectl apply -f trident-protect-snapshot-restore-cr.yaml
```

Use o CLI

Passos

1. Restaure o snapshot para um namespace diferente, substituindo os valores entre colchetes pelas informações do seu ambiente.

- O `snapshot` argumento usa um namespace e um nome de snapshot no formato `<namespace>/<name>`.
- O `namespace-mapping` argumento usa namespaces separados por dois pontos para mapear os 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>
```

Restaurar 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 o tempo de expiração do token de sessão da AWS seja suficiente para quaisquer operações de restauração do s3 de longa duração. Se o token expirar durante a operação de restauração, a operação pode falhar.

- Consulte o "Documentação do AWS API" para mais informações sobre como verificar a expiração do token de sessão atual.
- Consulte "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 que você criou, configure os seguintes atributos:
   - **metadata.name**: (*Obrigatório*) O nome deste recurso personalizado; escolha um nome único e adequado ao seu ambiente.
   - **spec.appVaultRef**: (*Obrigatório*) O nome do AppVault onde o conteúdo do snapshot está armazenado.
   - **spec.appArchivePath**: O caminho dentro do AppVault onde o conteúdo do snapshot está armazenado. Você pode usar o seguinte comando para encontrar esse 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
     ```

3. (*Opcional*) Se precisar selecionar apenas determinados recursos do aplicativo para restaurar, adicione filtros que incluam ou excluam recursos marcados com rótulos específicos:

   > 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 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. Após preencher o arquivo `trident-protect-snapshot-ipr-cr.yaml` com os valores corretos, aplique a CR:

```
kubectl apply -f trident-protect-snapshot-ipr-cr.yaml
```

Use o CLI

Passos

1. Restaure o snapshot para o namespace original, substituindo os valores entre colchetes pelas 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, foi concluída ou falhou.

Passos

1. Use o seguinte comando para recuperar o status da operação de restauração, substituindo os valores entre colchetes pelas informações do seu ambiente:

```
kubectl get backuprestore -n <namespace_name> <my_restore_cr_name> -o
jsonpath='{.status}'
```

Use as configurações avançadas de restauração do Trident Protect

Você pode personalizar as operações de restauração usando configurações avançadas, como anotações, configurações de namespace e opções de armazenamento para atender às suas necessidades específicas.

Anotações e rótulos de namespace durante operações de restauração e failover

Durante as operações de restauração e failover, os rótulos e anotações no namespace de destino são ajustados 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 já existentes são sobrescritos para corresponder ao valor do namespace de origem. Rótulos ou anotações que existem apenas no namespace de destino permanecem inalterados.

> Se você usa Red Hat OpenShift, é importante observar o papel crucial das anotações de namespace em ambientes OpenShift. As anotações de namespace garantem que os pods restaurados sigam 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 "OpenShift security context constraints documentação".

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
```

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 "Configurar configurações adicionais do helm chart do Trident Protect".

Se você instalou o aplicativo de origem usando Helm com a `--create-namespace` flag, um tratamento especial é dado à chave de rótulo `name`. Durante o processo de restauração ou failover, 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 é copiado para o namespace de destino sem alterações.

Exemplo

O exemplo a seguir apresenta um namespace de origem e um de 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 os rótulos são combinados ou sobrescritos no namespace de destino.

Antes da operação de restauração ou failover

A tabela a seguir ilustra o estado dos namespaces de origem e destino do exemplo antes da operação de restauração ou failover:

Espaço de nomes	Anotações	Etiquetas
Namespace ns-1 (fonte)	• annotation.one/key: "valoratualizado" • anotação.dois/chave: "true"	• ambiente=produção • compliance=hipaa • name=ns-1
Espaço de nomes ns-2 (destino)	• annotation.one/key: "true" • anotação.three/chave: "falso"	• role=database

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, outras foram sobrescritas e o `name` rótulo foi atualizado para corresponder ao namespace de destino:

Espaço de nomes	Anotações	Etiquetas
Espaço de nomes ns-2 (destino)	• annotation.one/key: "valoratualizado" • anotação.dois/chave: "true" • anotação.three/chave: "falso"	• name=ns-2 • compliance=hipaa • ambiente=produção • role=database

Campos suportados

Esta seção descreve os campos adicionais disponíveis para operações de restauração.

Mapeamento de classe de armazenamento

O `spec.storageClassMapping` atributo define um mapeamento de uma classe de armazenamento presente na aplicação de origem para uma nova classe de armazenamento no cluster de destino. Você pode usar isso ao migrar aplicações entre clusters com classes de armazenamento diferentes 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	string	O tempo máximo (em segundos) permitido para a operação de movimentação de dados ficar parada.	"300"
protect.trident.netapp.io/kopia-content-cache-size-limit-mb	string	O limite máximo de tamanho (em megabytes) para o cache de conteúdo do Kopia.	"1000"
protect.trident.netapp.io/pvc-bind-timeout-sec	string	Tempo máximo (em segundos) de espera para que qualquer PersistentVolumeClaims (PVC) recém-criado atinja a `Bound` fase antes que a operação falhe. Aplica-se a todos os tipos de CR de restauração (BackupRestore, BackupInplaceRestore, SnapshotRestore, SnapshotInplaceRestore). Use um valor maior se o seu backend de armazenamento ou cluster exigir mais tempo com frequência.	"1200" (20 minutos)

Replique aplicações usando NetApp SnapMirror e Trident Protect

Com 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, os rótulos e anotações no namespace de destino são ajustados 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 já existentes são sobrescritos para corresponder ao valor do namespace de origem. Rótulos ou anotações que existem apenas no namespace de destino permanecem inalterados.

> Se você usa Red Hat OpenShift, é importante observar o papel crucial das anotações de namespace em ambientes OpenShift. As anotações de namespace garantem que os pods restaurados sigam 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 "OpenShift security context constraints documentação".

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
```

> 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 "Configurar configurações adicionais do helm chart do Trident Protect".

Se você instalou o aplicativo de origem usando Helm com a `--create-namespace` flag, um tratamento especial é dado à chave de rótulo `name`. Durante o processo de restauração ou failover, 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 é copiado para o namespace de destino sem alterações.

Exemplo

O exemplo a seguir apresenta um namespace de origem e um de 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 os rótulos são combinados ou sobrescritos no namespace de destino.

Antes da operação de restauração ou failover

A tabela a seguir ilustra o estado dos namespaces de origem e destino do exemplo antes da operação de restauração ou failover:

Espaço de nomes	Anotações	Etiquetas
Namespace ns-1 (fonte)	• annotation.one/key: "valoratualizado" • anotação.dois/chave: "true"	• ambiente=produção • compliance=hipaa • name=ns-1
Espaço de nomes ns-2 (destino)	• annotation.one/key: "true" • anotação.three/chave: "falso"	• role=database

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, outras foram sobrescritas e o `name` rótulo foi atualizado para corresponder ao namespace de destino:

Espaço de nomes	Anotações	Etiquetas
Espaço de nomes ns-2 (destino)	• annotation.one/key: "valoratualizado" • anotação.dois/chave: "true" • anotação.three/chave: "falso"	• name=ns-2 • compliance=hipaa • ambiente=produção • role=database

Você pode configurar 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 Trident Protect".

Ganchos de execução durante failover e operações de reversão

Ao usar o relacionamento AppMirror para proteger sua aplicação, existem comportamentos específicos relacionados aos ganchos de execução que você deve conhecer durante operações de failover e reversão.

- Durante o failover, os hooks de execução são copiados automaticamente do cluster de origem para o cluster de destino. Você não precisa recriá-los manualmente. Após o failover, os hooks de execução estão presentes na aplicação e serão executados durante quaisquer ações relevantes.
- Durante a operação reversa ou ressincronização reversa, todos os ganchos de execução existentes no aplicativo são removidos. Quando o aplicativo de origem se torna o aplicativo de destino, esses ganchos de execução perdem a validade e são excluídos para impedir sua execução.

Para saber mais sobre ganchos de execução, consulte "Gerenciar os ganchos de execução do Trident Protect".

Estabeleça uma relação de replicação

Configurar uma relação de replicação envolve o seguinte:

- Escolher com que frequência você quer que o Trident Protect tire 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)

- Escolhendo o agendamento de replicação (inclui recursos do Kubernetes, bem como dados de volume persistente)
- Definindo o horário para a criação do snapshot

Passos

1. No cluster de origem, crie um AppVault para o aplicativo de origem. Dependendo do seu provedor de armazenamento, modifique um exemplo em "Recursos personalizados do AppVault" para adequá-lo ao seu ambiente:

Crie um AppVault usando um CR

a. Crie o arquivo de recurso personalizado (CR) e dê um nome a ele (por exemplo, `trident-protect-appvault-primary-source.yaml`).

b. Configurar os seguintes atributos:

- **metadata.name**: (*Obrigatório*) O nome do recurso personalizado AppVault. Anote o nome escolhido, pois outros arquivos CR necessários para um relacionamento de replicação fazem referência a esse valor.
- **spec.providerConfig**: (*Obrigatório*) 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 escolhidos, pois outros arquivos CR necessários para um relacionamento de replicação fazem referência a esses valores. Consulte "Recursos personalizados do AppVault" para exemplos de AppVault CRs com outros provedores.
- **spec.providerCredentials**: (*Obrigatório*) Armazena referências a quaisquer credencial necessárias para acessar o AppVault usando o provedor especificado.
  - **spec.providerCredentials.valueFromSecret**: (*Obrigatório*) Indica que o valor da credencial deve vir de um segredo.
    - **chave**: (*Obrigatório*) A chave válida do segredo a ser selecionada.
    - **nome**: (*Obrigatório*) Nome do segredo que contém o valor deste campo. Deve estar no mesmo namespace.
  - **spec.providerCredentials.secretAccessKey**: (*Obrigatório*) A chave de acesso usada para acessar o provedor. O **name** deve corresponder a **spec.providerCredentials.valueFromSecret.name**.
- **spec.providerType**: (*Obrigatório*) 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. Após preencher o arquivo `trident-protect-appvault-primary-source.yaml` com os valores corretos, aplique a CR:

```
kubectl apply -f trident-protect-appvault-primary-source.yaml -n
trident-protect
```

Crie um AppVault usando a CLI

a. Crie o AppVault, substituindo os valores entre colchetes 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 o CR do aplicativo de origem:

Crie o aplicativo de origem usando um CR

a. Crie o arquivo de recurso personalizado (CR) e dê um nome a ele (por exemplo, `trident-protect-app-source.yaml`).

b. Configurar os seguintes atributos:

- **metadata.name**: (*Obrigatório*) O nome do recurso personalizado do aplicativo. Anote o nome escolhido, pois outros arquivos CR necessários para um relacionamento de replicação fazem referência a esse valor.
- **spec.includedNamespaces**: (*Obrigatório*) Uma matriz 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 do aplicativo deve fazer parte desta matriz.

  Exemplo de 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. Após preencher o arquivo `trident-protect-app-source.yaml` com os valores corretos, aplique a CR:

```
kubectl apply -f trident-protect-app-source.yaml -n my-app-namespace
```

Crie o aplicativo de origem usando a CLI

a. Crie a aplicação 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. Esse snapshot é usado como base para o aplicativo no cluster de destino. Se você ignorar esta etapa, precisará aguardar a próxima execução de snapshot agendado para ter um snapshot recente. Para criar um snapshot sob demanda, consulte "Criar um snapshot sob demanda".

4. No cluster de origem, crie o CR de agendamento de replicação:

Além da programação fornecida abaixo, recomenda-se criar uma programação diária de snapshots separada, com um período de retenção de 7 dias, para manter um snapshot comum entre clusters ONTAP interligados. Isso garante que os snapshots estejam 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 de reversão. Essa abordagem torna o processo de reversão mais rápido e eficiente porque apenas as alterações feitas desde o último snapshot serão transferidas, não todos os dados.

Se um cronograma existente para o aplicativo já atender aos requisitos de retenção desejados, não serão necessários cronogramas adicionais.

Crie a programação de replicação usando uma CR

a. Crie um agendamento de replicação para o aplicativo de origem:

i. Crie o arquivo de recurso personalizado (CR) e dê um nome a ele (por exemplo, `trident-protect-schedule.yaml`).

ii. Configurar os seguintes atributos:

- **metadata.name**: (*Obrigatório*) O nome do recurso personalizado de agendamento.
- **spec.appVaultRef**: (*Obrigatório*) Este valor deve corresponder ao campo metadata.name do AppVault da aplicação de origem.
- **spec.applicationRef**: (*Obrigatório*) Este valor deve corresponder ao campo metadata.name do CR da aplicação de origem.
- **spec.backupRetention**: (*Obrigatório*) Este campo é obrigatório e o valor deve ser definido como 0.
- **spec.enabled**: deve ser definido como verdadeiro.
- **spec.granularidade**: Deve ser definida como `Custom`.
- **spec.recurrenceRule**: Defina uma data de início em 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. Após preencher o arquivo `trident-protect-schedule.yaml` com os valores corretos, aplique a 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 AppVault CR de aplicativo de origem idêntico ao AppVault CR que você aplicou no cluster de origem e nomeie-o (por exemplo, `trident-protect-appvault-primary-destination.yaml`).
6. Aplique 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 provedor de armazenamento, modifique um exemplo em "Recursos personalizados do AppVault" para adequá-lo ao seu ambiente:
   a. Crie o arquivo de recurso personalizado (CR) e dê um nome a ele (por exemplo, `trident-protect-appvault-secondary-destination.yaml`).
   b. Configurar os seguintes atributos:
      - **metadata.name**: (*Obrigatório*) O nome do recurso personalizado AppVault. Anote o nome escolhido, pois outros arquivos CR necessários para um relacionamento de replicação fazem referência a esse valor.
      - **spec.providerConfig**: (*Obrigatório*) 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 escolhidos, pois outros arquivos CR necessários para um relacionamento de replicação fazem referência a esses valores. Consulte "Recursos personalizados do AppVault" para exemplos de AppVault CRs com outros provedores.
      - **spec.providerCredentials**: (*Obrigatório*) Armazena referências a quaisquer credencial necessárias para acessar o AppVault usando o provedor especificado.
        - **spec.providerCredentials.valueFromSecret**: (*Obrigatório*) Indica que o valor da credencial

deve vir de um segredo.

- **chave**: (*Obrigatório*) A chave válida do segredo a ser selecionada.
- **nome**: (*Obrigatório*) Nome do segredo que contém o valor deste campo. Deve estar no mesmo namespace.

- **spec.providerCredentials.secretAccessKey**: (*Obrigatório*) A chave de acesso usada para acessar o provedor. O **name** deve corresponder a **spec.providerCredentials.valueFromSecret.name**.

- **spec.providerType**: (*Obrigatório*) 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. Após preencher o arquivo `trident-protect-appvault-secondary-destination.yaml` com os valores corretos, aplique a CR:

```
kubectl apply -f trident-protect-appvault-secondary-destination.yaml
-n trident-protect
```

8. No cluster de destino, crie um arquivo CR AppMirrorRelationship.

> Ao usar uma CR, crie manualmente o namespace de destino antes de aplicar a CR. Trident Protect cria namespaces automaticamente apenas quando se usa a CLI.

Crie um AppMirrorRelationship usando um CR

a. Crie o arquivo de recurso personalizado (CR) e dê um nome a ele (por exemplo, `trident-protect-relationship.yaml`).

b. Configurar os seguintes atributos:

- **metadata.name:** (Obrigatório) O nome do recurso personalizado AppMirrorRelationship.
- **spec.destinationAppVaultRef**: (*Obrigatório*) Este valor deve corresponder ao nome do AppVault para o aplicativo de destino no cluster de destino.
- **spec.namespaceMapping**: (*Obrigatório*) Os namespaces de destino e de origem devem corresponder ao namespace do aplicativo definido no respectivo CR do aplicativo.
- **spec.sourceAppVaultRef**: (*Obrigatório*) Este valor deve corresponder ao nome do AppVault para o aplicativo de origem.
- **spec.sourceApplicationName**: (*Obrigatório*) Este valor deve corresponder ao nome do aplicativo de origem que você definiu 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 estar vinculada a uma VM de armazenamento ONTAP que esteja emparelhada com o ambiente de origem. Se a classe de armazenamento não for fornecida, a classe de armazenamento padrão do cluster será usada por padrão.
- **spec.recurrenceRule**: Defina uma data de início em 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. Após preencher o arquivo `trident-protect-relationship.yaml` com os valores corretos, aplique a 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 por 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> -n <application_namespace>
```

Exemplo:

```
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. (*Opcional*) No cluster de destino, verifique o estado e o status da relação de replicação:

```
kubectl get amr -n my-app-namespace <relationship name> -o=jsonpath
='{.status}' | jq
```

Fail over para o cluster de destino

Com 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 app online no cluster de destino. Trident Protect não interrompe o app no cluster de origem se ele estiver operacional.

Passos

1. No cluster de destino, edite o arquivo CR AppMirrorRelationship (por exemplo, `trident-protect-relationship.yaml`) e altere o valor de **spec.desiredState** para `Promoted`.
2. Salve o arquivo CR.
3. Aplique o CR:

```
kubectl apply -f trident-protect-relationship.yaml -n my-app-namespace
```

4. (*Opcional*) Crie quaisquer cronogramas de proteção que você precise no aplicativo com failover.
5. (*Opcional*) Verifique o estado e o status da relação de replicação:

```
kubectl get amr -n my-app-namespace <relationship name> -o=jsonpath
='{.status}' | jq
```

Ressincronizar uma relação de replicação com failover

A operação de ressincronização restabelece a relação de replicação. Após a execução de uma operação de ressincronização, o aplicativo de origem original torna-se o aplicativo em execução e quaisquer alterações feitas no aplicativo em execução no cluster de destino são descartadas.

O processo interrompe o app no cluster de destino antes de restabelecer a replicação.

Quaisquer dados gravados no aplicativo de destino durante a transição de falha serão perdidos.

Passos

1. Opcional: No cluster de origem, crie um snapshot 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. Aplique o CR:

```
kubectl apply -f trident-protect-relationship.yaml -n my-app-namespace
```

5. Se você criou algum agendamento de proteção no cluster de destino para proteger o aplicativo em failover, remova-os. Quaisquer agendamentos restantes causam falhas no snapshot de volume.

Resincronizar reversamente uma relação de replicação com failover

Ao reverter a resincronização de uma relação de replicação com falha, o aplicativo de destino torna-se o aplicativo de origem e a origem torna-se o destino. As alterações feitas no aplicativo de destino durante a falha são mantidas.

Passos

1. No cluster de destino original, exclua o CR AppMirrorRelationship. Isso faz com que o destino se torne o cluster de origem. Se houver algum agendamento de proteção restante no novo cluster de destino, remova-os.
2. Estabeleça uma relação de replicação aplicando os arquivos CR que você usou originalmente para configurar a relação aos clusters opostos.
3. Certifique-se de que o novo destino (cluster de origem) esteja configurado com ambas as CRs AppVault.
4. Configure uma relação de replicação no cluster oposto, configurando valores para a direção inversa.

Inverter a direção da replicação do aplicativo

Ao inverter a direção da replicação, 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. Trident Protect interrompe a aplicação de origem e replica os dados para o destino antes de realizar o failover para a aplicação de destino.

Nessa situação, você está trocando o cluster de origem e o cluster de destino.

Passos

1. No cluster de origem, crie um snapshot de desligamento:

Crie um instantâneo de desligamento usando uma CR

a. Desative os agendamentos da política de proteção para o aplicativo de origem.

b. Crie um arquivo CR ShutdownSnapshot:

   i. Crie o arquivo de recurso personalizado (CR) e dê um nome a ele (por exemplo, `trident-protect-shutdownsnapshot.yaml`).

   ii. Configurar os seguintes atributos:

   - **metadata.name**: (*Obrigatório*) O nome do recurso personalizado.
   - **spec.AppVaultRef**: (*Obrigatório*) Este valor deve corresponder ao campo metadata.name do AppVault da aplicação de origem.
   - **spec.ApplicationRef**: (*Obrigatório*) Este valor deve corresponder ao campo metadata.name do arquivo CR do aplicativo 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. Após preencher o arquivo `trident-protect-shutdownsnapshot.yaml` com os valores corretos, aplique a CR:

```
kubectl apply -f trident-protect-shutdownsnapshot.yaml -n my-app-
namespace
```

Crie um Snapshot de desligamento usando a CLI

a. Crie o snapshot de desligamento, substituindo os valores entre colchetes pelas 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 snapshot de desligamento, obtenha o status do snapshot de desligamento:

```
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 chamada de nome base; isso será tudo o que vem depois da última barra):

```
k get shutdownsnapshot -n my-app-namespace <shutdown_snapshot_name> -o
jsonpath='{.status.appArchivePath}'
```

4. Realize um failover do novo cluster de destino para o novo cluster de origem, com a seguinte alteração:

> Na etapa 2 do procedimento de failover, inclua o campo `spec.promotedSnapshot` no arquivo CR AppMirrorRelationship e defina seu valor para o basename que você registrou na etapa 3 acima.

5. Execute as etapas de resincronização reversa em Resincronizar reversamente uma relação de replicação com failover.
6. Ative os agendamentos de proteção no novo cluster de origem.

Resultado

As seguintes ações ocorrem devido à replicação reversa:

- É feita uma snapshot dos recursos Kubernetes do aplicativo de origem original.
- Os pods do aplicativo de origem original são interrompidos corretamente ao excluir os recursos do Kubernetes do aplicativo (mantendo os PVCs e PVs no lugar).
- Após o desligamento dos pods, snapshots dos volumes do aplicativo são criados e replicados.
- As relações do SnapMirror são rompidas, tornando os volumes de destino prontos para leitura/gravação.
- Os recursos do Kubernetes do aplicativo são restaurados a partir do Snapshot anterior ao desligamento, usando os dados de volume replicados após o desligamento do aplicativo de origem original.
- A replicação é restabelecida na direção inversa.

Restaurar aplicativos para o cluster de origem

Com Trident Protect, você pode realizar o "fail back" após uma operação de failover seguindo a seguinte sequência de operações. Nesse fluxo de trabalho para restaurar a direção de replicação original, Trident Protect replica (ressincroniza) quaisquer alterações do aplicativo de volta para o aplicativo de origem original antes de reverter a direção da replicação.

Esse processo começa a partir de uma relação que concluiu um failover para um destino e envolve as seguintes etapas:

- Comece com um estado de failover.

- Reverter a resincronização da relação de replicação.

> Não execute uma operação de ressincronização normal, pois isso descartará dados gravados no cluster de destino durante o procedimento de fail over.

- Inverta a direção da replicação.

Passos

1. Execute os Resincronizar reversamente uma relação de replicação com failover passos.
2. Execute os Inverter a direção da replicação do aplicativo passos.

Excluir uma relação de replicação

Você pode excluir uma relação de replicação a qualquer momento. Ao excluir a relação de replicação do aplicativo, o resultado são dois aplicativos separados, sem nenhuma relação entre eles.

Passos

1. No cluster de destino atual, exclua o AppMirrorRelationship CR:

```
kubectl delete -f trident-protect-relationship.yaml -n my-app-namespace
```

Migrar aplicativos usando Trident Protect

Você pode migrar seus aplicativos entre clusters ou para diferentes classes de armazenamento restaurando dados de backup.

> Ao migrar um aplicativo, todos os ganchos de execução configurados para o aplicativo são migrados juntamente com o aplicativo. Se houver um gancho de execução pós-restauração, ele é executado automaticamente como parte da operação de restauração.

Operações de backup e restauração

Para executar operações de backup e restauração nos seguintes cenários, você pode automatizar tarefas específicas de backup e restauração.

Clonar para o mesmo cluster

Para clonar um aplicativo para o mesmo cluster, crie um snapshot ou backup e restaure os dados para o mesmo cluster.

Passos

1. Faça um dos seguintes procedimentos:
   a. "Criar um snapshot".
   b. "Criar um backup".
2. No mesmo cluster, faça um dos seguintes procedimentos, dependendo se você criou um snapshot ou um backup:
   a. "Restaure seus dados a partir do snapshot".

b. "Restaure seus dados do backup".

Clonar 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 Trident Protect esteja instalado no cluster de destino.

Você pode replicar um aplicativo entre diferentes clusters usando "SnapMirror replicação".

Passos

1. "Criar um backup".
2. Certifique-se de que o CR AppVault para o bucket do storage 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 armazenamento para outra classe de armazenamento

Você pode migrar aplicativos de uma classe de armazenamento para outra restaurando um backup na 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 snapshot usando um CR

Passos

1. Crie o arquivo de recurso personalizado (CR) e nomeie-o `trident-protect-snapshot-restore-cr.yaml`.
2. No arquivo que você criou, configure os seguintes atributos:
   - **metadata.name**: (*Obrigatório*) O nome deste recurso personalizado; escolha um nome único e adequado ao seu ambiente.
   - **spec.appArchivePath**: O caminho dentro do AppVault onde o conteúdo do snapshot está armazenado. Você pode usar o seguinte comando para encontrar esse caminho:

     ```
     kubectl get snapshots <my-snapshot-name> -n trident-protect -o jsonpath='{.status.appArchivePath}'
     ```

   - **spec.appVaultRef**: (*Obrigatório*) O nome do AppVault onde o conteúdo do snapshot está 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` pelas 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 precisar selecionar apenas determinados recursos do aplicativo para restaurar, adicione filtragem que inclua ou exclua recursos marcados com rótulos específicos:
   - **resourceFilter.resourceSelectionCriteria**: (Obrigatório 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 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 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. Após preencher o arquivo `trident-protect-snapshot-restore-cr.yaml` com os valores corretos, aplique a CR:

```
kubectl apply -f trident-protect-snapshot-restore-cr.yaml
```

Restaurar o snapshot usando a CLI

Passos

1. Restaure o snapshot para um namespace diferente, substituindo os valores entre colchetes pelas informações do seu ambiente.

   - O `snapshot` argumento usa um namespace e um nome de snapshot no formato `<namespace>/<name>`.
   - O `namespace-mapping` argumento usa namespaces separados por dois pontos para mapear os 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>
```

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, pode usar um gancho de execução para pausar todas as transações do banco de dados antes de um Snapshot e retomá-las após a conclusão do Snapshot. Isso garante Snapshots consistentes para o aplicativo.

Tipos de ganchos de execução

Trident Protect suporta os seguintes tipos de ganchos de execução, com base em quando podem ser executados:

- Pré-Snapshot
- 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. Quaisquer hooks de execução pré-operacionais personalizados aplicáveis são executados nos contêineres apropriados. Você pode criar e executar quantos hooks de pré-operação personalizados forem necessários, mas a ordem de execução desses hooks antes da operação não é garantida nem configurável.
2. Ocorrem congelamentos do sistema de arquivos, se aplicável. "Saiba mais sobre como configurar o congelamento do sistema de arquivos com Trident Protect".
3. A operação de proteção de dados é realizada.
4. Os sistemas de arquivos congelados são descongelados, se aplicável.
5. Quaisquer hooks de execução pós-operação personalizados aplicáveis são executados nos contêineres apropriados. Você pode criar e executar quantos hooks de pós-operação personalizados forem necessários, mas a ordem de execução desses hooks 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 é garantida. No entanto, a ordem de execução de ganchos de tipos diferentes é garantida. Por exemplo, a seguir está a ordem de execução de uma configuração que possui todos os

diferentes tipos de ganchos:

1. Ganchos pré-Snapshot executados
2. Ganchos pós-Snapshot executados
3. Ganchos de pré-backup executados
4. Ganchos pós-backup executados

O exemplo de ordem anterior só se aplica quando você executa um backup que não usa um snapshot existente.

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 os scripts convenientemente. Depois de habilitar os ganchos de execução em um ambiente de produção, teste os snapshots e backups resultantes para garantir que estejam consistentes. Você pode fazer isso clonando o app para um namespace temporário, restaurando o snapshot ou backup e, em seguida, testando o app.

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.

Observações importantes sobre ganchos de execução personalizados

Considere o seguinte ao planejar os ganchos de execução para seus aplicativos.

- Um gancho de execução deve usar um script para executar ações. Vários ganchos de execução podem referenciar o mesmo script.
- Trident Protect exige que os scripts que os ganchos de execução usam sejam escritos no formato de scripts shell executáveis.
- O tamanho do script está limitado a 96KB.
- 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.

Como os hooks de execução frequentemente reduzem ou desativam completamente a funcionalidade do aplicativo em que estão sendo executados, você deve sempre tentar minimizar o tempo que seus hooks de execução personalizados levam para serem executados. Se você iniciar uma operação de backup ou snapshot com hooks de execução associados, mas depois cancelá-la, os hooks ainda poderão ser executados se a operação de backup ou snapshot já tiver sido iniciada. Isso significa que a lógica usada em um hook de execução pós-backup não pode presumir que o backup foi concluído.

Filtros de gancho de execução

Ao adicionar ou editar um gancho de execução para um aplicativo, você pode adicionar filtros ao gancho de execução para gerenciar quais contêineres o gancho irá corresponder. Os filtros são úteis para aplicativos que usam a mesma imagem de contêiner em todos os contêineres, mas podem usar cada imagem para uma finalidade diferente (como Elasticsearch). Os filtros permitem criar cenários em que ganchos de execução são executados em alguns, mas não necessariamente em todos os contêineres idênticos. Se você criar vários filtros para um único gancho de execução, eles serão combinados com um operador lógico AND. 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 contêineres no seu cluster. Quando um gancho corresponde a um contêiner, o gancho executará seu script associado nesse contêiner. As expressões regulares para filtros usam a sintaxe Regular Expression 2 (RE2), que não permite criar um filtro que exclua contêineres da lista de correspondências. Para obter informações sobre a sintaxe que o Trident Protect oferece suporte para expressões regulares em filtros de ganchos de execução, consulte "Suporte à sintaxe de Regular Expression 2 (RE2)".

> Se você adicionar um filtro de namespace a um gancho de execução que é executado após uma operação de restauração ou clonagem e a origem e o destino estiverem em namespaces diferentes, o filtro de namespace será aplicado apenas ao namespace de destino.

Exemplos de ganchos de execução

Visite o "Projeto NetApp Verda GitHub" para baixar hooks de execução reais para aplicativos populares como Apache Cassandra e Elasticsearch. Você também pode ver exemplos e obter ideias para estruturar seus próprios hooks de execução personalizados.

Crie um gancho de execução

Você pode criar um gancho de execução personalizado para um aplicativo usando 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**: (*Obrigatório*) O nome deste recurso personalizado; escolha um nome único e adequado ao seu ambiente.
   - **spec.applicationRef**: (*Obrigatório*) O nome do aplicativo Kubernetes para o qual o gancho de execução será executado.
   - **spec.stage**: (*Obrigatório*) Uma string indicando em qual estágio da ação o gancho de execução deve ser executado. Valores possíveis:
     - Pré
     - Publicar
   - **spec.action**: (*Obrigatório*) Uma string indicando qual ação o gancho de execução executará, assumindo que quaisquer filtros de gancho de execução especificados sejam correspondidos. Valores possíveis:
     - Snapshot
     - Backup
     - Restaurar
     - Failover
   - **spec.enabled**: (opcional) Indica se este gancho de execução está habilitado ou desabilitado. Se não for especificado, o valor padrão é true.
   - **spec.hookSource**: (*Obrigatório*) Uma string contendo o script de gancho codificado em base64.
   - **spec.timeout**: (Opcional) Um número que define por quanto tempo, em minutos, o gancho de execução pode ser executado. O valor mínimo é 1 minuto e o valor padrão é 25 minutos se não for especificado.
   - **spec.arguments**: (opcional) Uma lista YAML de argumentos que você pode especificar para o gancho de execução.
   - **spec.matchingCriteria**: (*Opcional*) Uma lista opcional de pares de chave-valor de critérios, cada par constituindo um filtro de gancho de execução. Você pode adicionar até 10 filtros por gancho de execução.
   - **spec.matchingCriteria.type**: (Opcional) Uma string que identifica o tipo de filtro do gancho de execução. Valores possíveis:
     - ContainerImage
     - ContainerName
     - PodName
     - PodLabel
     - NamespaceName
   - **spec.matchingCriteria.value**: (*Opcional*) Uma string ou expressão regular que identifica 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. Após preencher o arquivo CR com os valores corretos, aplique o CR:

```
kubectl apply -f trident-protect-hook.yaml
```

Use o CLI

Passos

1. Crie o gancho de execução, substituindo os valores entre colchetes 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>
```

Executar manualmente um gancho de execução

Você pode executar um gancho de execução manualmente para fins de teste ou se precisar executá-lo novamente após uma falha. Você precisa ter permissões de Proprietário, Administrador ou Membro para executar ganchos de execução manualmente.

A execução manual de um gancho de execução consiste em duas etapas básicas:

1. Crie um backup de recurso, que coleta recursos e cria um backup deles, determinando onde o gancho será executado
2. Execute o hook de execução no backup

Passo 1: Crie um backup de recurso

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**: (*Obrigatório*) O nome deste recurso personalizado; escolha um nome único e adequado ao seu ambiente.
   - **spec.applicationRef**: (*Obrigatório*) O nome do Kubernetes da aplicação para a qual criar o backup do recurso.
   - **spec.appVaultRef**: (*Obrigatório*) O nome do AppVault onde o conteúdo do backup está armazenado.
   - **spec.appArchivePath**: O caminho dentro do AppVault onde o conteúdo do backup está armazenado. Você pode usar o seguinte comando para encontrar esse 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. Após preencher o arquivo CR com os valores corretos, aplique o CR:

```
kubectl apply -f trident-protect-resource-backup.yaml
```

Use o CLI

Passos

1. Crie o backup, substituindo os valores entre colchetes pelas 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. Veja o status do backup. Você pode usar este comando de exemplo repetidamente até que a operação seja 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>
```

Etapa 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**: (*Obrigatório*) O nome deste recurso personalizado; escolha um nome único e adequado ao seu ambiente.
   - **spec.applicationRef**: (*Obrigatório*) Certifique-se de que este valor corresponda ao nome do aplicativo da ResourceBackup CR que você criou na etapa 1.
   - **spec.appVaultRef**: (*Obrigatório*) Certifique-se de que este valor corresponda ao appVaultRef do ResourceBackup CR que você criou na etapa 1.
   - **spec.appArchivePath**: Certifique-se de que este valor corresponda ao appArchivePath do ResourceBackup CR que você criou na etapa 1.

     ```
     kubectl get backups <BACKUP_NAME> -n my-app-namespace -o jsonpath='{.status.appArchivePath}'
     ```

   - **spec.action**: (*Obrigatório*) Uma string indicando qual ação o gancho de execução executará, assumindo que quaisquer filtros de gancho de execução especificados sejam correspondidos. Valores possíveis:
     - Snapshot
     - Backup
     - Restaurar
     - Failover
   - **spec.stage**: (*Obrigatório*) Uma string indicando em qual estágio da ação o gancho de execução deve ser executado. Essa execução de gancho não executará ganchos em nenhum outro estágio. Valores possíveis:
     - Pré
     - Publicar

     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. Após preencher o arquivo CR com os valores corretos, aplique o CR:

```
kubectl apply -f trident-protect-hook-run.yaml
```

Use o CLI

Passos

1. Crie a solicitação de execução manual do hook:

```
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 gancho. Você pode executar este comando repetidamente até que a operação seja concluída:

```
tridentctl protect get exechooksrun -n <my_app_namespace>
<my_exec_hook_run_name>
```

3. Descreva o objeto exechooksrun para ver os detalhes finais e o status:

```
kubectl -n <my_app_namespace> describe exechooksrun
<my_exec_hook_run_name>
```

Desinstale Trident Protect

Você pode precisar remover componentes do Trident Protect se estiver atualizando de uma versão de avaliação para uma versão completa do produto.

Para remover Trident Protect, execute as seguintes etapas.

Passos

1. Remova os arquivos CR do Trident Protect:

   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
   ```

Informações sobre direitos autorais

Informações sobre marcas comerciais