Proteja seus aplicativos com o Trident Protect.

Trident

NetApp
February 20, 2026

Índice

Proteja seus aplicativos com o Trident Protect.

Saiba mais sobre o Trident Protect.

O NetApp Trident Protect oferece recursos avançados de gerenciamento de dados de aplicativos que aprimoram a funcionalidade e a disponibilidade de aplicativos Kubernetes com estado, com suporte dos sistemas de armazenamento NetApp ONTAP e do provisionador de armazenamento NetApp Trident CSI. O Trident Protect simplifica o gerenciamento, a proteção e a movimentação de cargas de trabalho conteinerizadas em nuvens públicas e ambientes locais. Ele também oferece recursos de automação por meio de sua API e CLI.

Você pode proteger aplicativos com o Trident Protect criando recursos personalizados (CRs) ou usando a CLI do Trident Protect.

O que se segue?

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

- "Requisitos do Trident Protect"

Instale o Trident Protect

Requisitos do Trident Protect

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

Compatibilidade do cluster Kubernetes com o Trident Protect

O Trident Protect é compatível com uma ampla gama de ofertas de Kubernetes totalmente gerenciadas e autogerenciadas, incluindo:

- Amazon Elastic Kubernetes Service (EKS)
- Google Kubernetes Engine (GKE)
- Microsoft Azure Kubernetes Service (AKS)
- Red Hat OpenShift
- SUSE Rancher
- Portfólio do VMware Tanzu
- Kubernetes upstream

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

Compatibilidade com o backend de armazenamento Trident Protect

O Trident Protect é compatível com os seguintes sistemas de armazenamento:

- Amazon FSX para NetApp ONTAP
- Cloud Volumes ONTAP
- Storage arrays ONTAP
- Google Cloud NetApp volumes
- Azure NetApp Files

Certifique-se de que o back-end de storage atenda aos seguintes requisitos:

- Certifique-se de que o armazenamento NetApp conectado ao cluster esteja usando o Trident 24.02 ou mais recente (Trident 24.10 é recomendado).
- Verifique se você tem um back-end de storage do NetApp ONTAP.
- Certifique-se de ter configurado um bucket de armazenamento de objetos para armazenar backups.
- Crie todos os namespaces de aplicativos que você planeja usar para aplicativos ou operações de gerenciamento de dados de aplicativos. O Trident Protect não cria esses namespaces para você; se você especificar um namespace inexistente em um recurso personalizado, a operação falhará.

Requisitos para volumes nas-economia

O Trident Protect oferece suporte a operações de backup e restauração para volumes NAS Economy. Atualmente, não há suporte para snapshots, clones e replicação do SnapMirror para volumes nas-economy. Você precisa habilitar um diretório de snapshots para cada volume nas-economy que planeja usar com o Trident Protect.

Alguns aplicativos não são compatíveis com volumes que usam um diretório instantâneo. Para esses aplicativos, você precisa ocultar o diretório instantâneo executando o seguinte comando no sistema de armazenamento ONTAP:

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

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

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

Você pode habilitar diretórios de snapshot por padrão para novos volumes definindo a opção de configuração de back-end do Trident `snapshotDir` como `true`. Os volumes existentes não são afetados.

Proteção de dados com máquinas virtuais do KubeVirt

O Trident Protect oferece recursos de congelamento e descongelamento do sistema de arquivos para máquinas virtuais KubeVirt durante operações de proteção de dados, garantindo a consistência dos dados. O método de configuração e o comportamento padrão para operações de congelamento de máquinas virtuais variam entre as versões do Trident Protect, sendo que as versões mais recentes oferecem configuração simplificada por meio de parâmetros do gráfico Helm.

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

Trident Protect 25.10 e versões mais recentes

O Trident Protect congela e descongela automaticamente os sistemas de arquivos do KubeVirt durante as operações de proteção de dados para garantir a consistência. A partir do Trident Protect 25.10, você pode desativar esse comportamento usando o `vm.freeze` parâmetro durante a instalação do gráfico Helm. O parâmetro está ativado por padrão.

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

Trident Protect 24.10.1 a 25.06

A partir da versão 24.10.1 do Trident Protect, o sistema congela e descongela automaticamente os sistemas de arquivos do KubeVirt durante as operações de proteção de dados. Opcionalmente, você pode desativar esse comportamento automático usando o seguinte comando:

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

Trident Protect 24.10

O Trident Protect 24.10 não garante automaticamente um estado consistente para os sistemas de arquivos das VMs do KubeVirt durante as operações de proteção de dados. Se você deseja proteger os dados da sua máquina virtual KubeVirt usando o Trident Protect 24.10, precisa habilitar manualmente a funcionalidade de congelamento/descongelamento dos sistemas de arquivos antes da operação de proteção de dados. Isso garante que os sistemas de arquivos estejam em um estado consistente.

Você pode configurar o Trident Protect 24.10 para gerenciar o congelamento e o descongelamento do sistema de arquivos da VM durante as operações de proteção de dados."configuração da virtualização" e então usando o seguinte comando:

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

Requisitos para replicação do SnapMirror

A replicação NetApp SnapMirror está disponível para uso com o Trident Protect para as seguintes soluções ONTAP :

- Sistemas NetApp FAS, AFF e ASA locais. A replicação SnapMirror com Trident protect não é atualmente suportada para sistemas ASA r2.
- NetApp ONTAP Select
- NetApp Cloud Volumes ONTAP
- Amazon FSX para NetApp ONTAP

Requisitos de cluster do ONTAP para replicação do SnapMirror

Se você planeja usar a replicação do SnapMirror, verifique se o cluster do ONTAP atende aos seguintes requisitos:

- * NetApp Trident*: O NetApp Trident deve estar presente nos clusters Kubernetes de origem e destino que utilizam o ONTAP como backend. O Trident Protect oferece suporte à replicação com a tecnologia NetApp SnapMirror usando classes de armazenamento com suporte dos seguintes drivers:
  - `ontap-nas` : NFS
  - `ontap-san` : iSCSI
  - `ontap-san` :FC
  - `ontap-san` : NVMe/TCP (requer no mínimo a versão ONTAP 9.15.1)
- **Licenças**: As licenças assíncronas do ONTAP SnapMirror usando o pacote proteção de dados devem estar ativadas nos clusters ONTAP de origem e destino. "Visão geral do licenciamento do SnapMirror no ONTAP"Consulte para obter mais informações.

  A partir do ONTAP 9.10,1, todas as licenças são entregues como um arquivo de licença NetApp (NLF), que é um único arquivo que permite vários recursos. "Licenças incluídas no ONTAP One"Consulte para obter mais informações.

  Somente a proteção assíncrona SnapMirror é suportada.

Considerações de peering para replicação do SnapMirror

Certifique-se de que seu ambiente atenda aos seguintes requisitos se você planeja usar peering de back-end de storage:

- **Cluster e SVM**: Os backends de storage do ONTAP devem ser colocados em Contato. "Visão geral do peering de cluster e SVM"Consulte para obter mais informações.

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

- **NetApp Trident e SVM**: Os SVMs remotos pareados devem estar disponíveis para o NetApp Trident no cluster de destino.
- **Backends gerenciados**: Você precisa adicionar e gerenciar backends de armazenamento ONTAP no Trident Protect para criar uma relação de replicação.

Configuração Trident / ONTAP para replicação SnapMirror

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

Requisitos de cluster do Kubernetes para replicação do SnapMirror

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

- **Acessibilidade do AppVault**: Os clusters de origem e destino devem ter acesso à rede para ler e gravar no AppVault para replicação de objetos do aplicativo.
- **Conectividade de rede**: configure regras de firewall, permissões de bucket e listas de permissões de IP para permitir a comunicação entre os dois clusters e o AppVault através de WANs.

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

Instale e configure o Trident Protect.

Se o seu ambiente atender aos requisitos do Trident Protect, você pode seguir estas etapas para instalar o Trident Protect em seu cluster. Você pode obter o Trident Protect da NetApp ou instalá-lo a partir do seu próprio registro privado. A instalação a partir de um registro privado é útil caso seu cluster não tenha acesso à Internet.

Instale o Trident Protect

Instale o Trident Protect da NetApp.

Passos

1. Adicione o repositório Helm do Trident:

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

2. Use o Helm para instalar o Trident Protect. Substituir `<name-of-cluster>` com um nome de cluster, que será atribuído ao cluster e usado para identificar os backups e snapshots do cluster:

```
helm install trident-protect netapp-trident-protect/trident-protect
--set clusterName=<name-of-cluster> --version 100.2510.0 --create
-namespace --namespace trident-protect
```

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

```
helm install trident-protect netapp-trident-protect/trident-protect
--set clusterName=<name-of-cluster> --set logLevel=debug --version
100.2510.0 --create-namespace --namespace trident-protect
```

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

Instale o Trident Protect a partir de um registro privado.

Você pode instalar o Trident Protect a partir de um registro de imagens privado caso seu cluster Kubernetes não tenha acesso à Internet. Nestes exemplos, substitua os valores entre colchetes por informações do seu ambiente:

Passos

1. Puxe as seguintes imagens para a sua máquina local, atualize as etiquetas e, em seguida, envie-as para o seu registo privado:

```
docker.io/netapp/controller:25.10.0
docker.io/netapp/restic:25.10.0
docker.io/netapp/kopia:25.10.0
docker.io/netapp/kopiablockrestore:25.10.0
docker.io/netapp/trident-autosupport:25.10.0
docker.io/netapp/exechook:25.10.0
docker.io/netapp/resourcebackup:25.10.0
docker.io/netapp/resourcerestore:25.10.0
docker.io/netapp/resourcedelete:25.10.0
docker.io/netapp/trident-protect-utils:v1.0.0
```

Por exemplo:

```
docker pull docker.io/netapp/controller:25.10.0
```

```
docker tag docker.io/netapp/controller:25.10.0 <private-registry-url>/controller:25.10.0
```

```
docker push <private-registry-url>/controller:25.10.0
```

> Para obter o gráfico Helm, primeiro faça o download do gráfico Helm em um computador com acesso à internet usando `helm pull trident-protect --version 100.2510.0 --repo https://netapp.github.io/trident-protect-helm-chart`Em seguida, copie o resultado. `trident-protect-100.2510.0.tgz` copie o arquivo para seu ambiente offline e instale-o usando `helm install trident-protect ./trident-protect-100.2510.0.tgz` em vez da referência ao repositório na etapa final.

2. Crie o namespace do sistema Trident Protect:

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

3. Inicie sessão no registo:

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

4. Crie um segredo para usar para autenticação de Registro privado:

```
kubectl create secret docker-registry regcred --docker
-username=<registry-username> --docker-password=<api-token> -n
trident-protect --docker-server=<private-registry-url>
```

5. Adicione o repositório Helm do Trident:

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

6. Crie um arquivo chamado `protectValues.yaml` . Certifique-se de que contenha as seguintes configurações do Trident Protect:

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

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

7. Use o Helm para instalar o Trident Protect. Substituir `<name_of_cluster>` com um nome de cluster, que será atribuído ao cluster e usado para identificar os backups e snapshots do cluster:

```
helm install trident-protect netapp-trident-protect/trident-protect
--set clusterName=<name_of_cluster> --version 100.2510.0 --create
-namespace --namespace trident-protect -f protectValues.yaml
```

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

```
helm install trident-protect netapp-trident-protect/trident-protect
--set clusterName=<name-of-cluster> --set logLevel=debug --version
100.2510.0 --create-namespace --namespace trident-protect -f
protectValues.yaml
```

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

Para opções adicionais de configuração do gráfico Helm, incluindo configurações de AutoSupport e filtragem de namespace, consulte "Personalize a instalação do Trident Protect".

Instale o plugin Trident Protect CLI

Você pode usar o plugin de linha de comando Trident Protect, que é uma extensão do Trident. `tridentctl` Utilitário para criar e interagir com recursos personalizados (CRs) do Trident Protect.

Instale o plugin Trident Protect CLI

Antes de usar o utilitário de linha de comando, você precisa instalá-lo na máquina usada para acessar o cluster. Siga estes passos, dependendo se a sua máquina utiliza uma CPU x64 ou ARM.

Faça o download do plugin para CPUs Linux AMD64

Passos

1. Baixe o plugin Trident Protect CLI:

```
curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-protect/releases/download/25.10.0/tridentctl-protect-linux-amd64
```

Faça o download do plugin para CPUs Linux ARM64

Passos

1. Baixe o plugin Trident Protect CLI:

```
curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-protect/releases/download/25.10.0/tridentctl-protect-linux-arm64
```

Baixe o plugin para CPUs Mac AMD64

Passos

1. Baixe o plugin Trident Protect CLI:

```
curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-protect/releases/download/25.10.0/tridentctl-protect-macos-amd64
```

Baixe o plugin para CPUs Mac ARM64

Passos

1. Baixe o plugin Trident Protect CLI:

```
curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-protect/releases/download/25.10.0/tridentctl-protect-macos-arm64
```

1. Ativar permissões de execução para o binário do plugin:

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

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

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

3. Opcionalmente, você pode copiar o binário do plugin para um local em seu diretório home. Neste caso, é recomendável garantir que a localização faça parte da variável PATH:

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

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

Veja a ajuda do plugin Trident CLI

Você pode usar os recursos integrados de ajuda do plugin para obter ajuda detalhada sobre os recursos do plugin:

Passos

1. Utilize a função de ajuda para visualizar as orientações de utilização:

```
tridentctl-protect help
```

Ativar a auto-conclusão do comando

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

Ative a auto-conclusão para o shell Bash

Passos

1. Crie o script de conclusão:

```
tridentctl-protect completion bash > tridentctl-completion.bash
```

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

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

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

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

4. Adicione a seguinte linha ao `~/.bashrc` arquivo em seu diretório inicial:

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

Ative a auto-conclusão para o shell Z.

Passos

1. Crie o script de conclusão:

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

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

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

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

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

4. Adicione a seguinte linha ao `~/.zprofile` arquivo em seu diretório inicial:

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

Resultado

Após o seu próximo login shell, você pode usar o comando auto-completação com o plugin tridentctl-protect.

Personalize a instalação do Trident Protect

Você pode personalizar a configuração padrão do Trident Protect para atender aos requisitos específicos do seu ambiente.

Especifique os limites de recursos do contêiner Trident Protect.

Você pode usar um arquivo de configuração para especificar limites de recursos para os contêineres do Trident Protect após a instalação do Trident Protect. A definição de limites de recursos permite controlar a quantidade de recursos do cluster que são consumidos pelas operações do Trident Protect.

Passos

1. Crie um arquivo chamado `resourceLimits.yaml`.
2. Preencha o arquivo com as opções de limite de recursos para os contêineres do Trident Protect de acordo com as necessidades do seu ambiente.

   O seguinte exemplo de arquivo de configuração mostra as configurações disponíveis e contém os valores padrão para cada limite de recursos:

```
---
jobResources:
  defaults:
    limits:
      cpu: 8000m
      memory: 10000Mi
      ephemeralStorage: ""
    requests:
      cpu: 100m
      memory: 100Mi
      ephemeralStorage: ""
  resticVolumeBackup:
    limits:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
    requests:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
  resticVolumeRestore:
    limits:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
```

```
    requests:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
  kopiaVolumeBackup:
    limits:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
    requests:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
  kopiaVolumeRestore:
    limits:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
    requests:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
```

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

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

Personalizar restrições de contexto de segurança

Você pode usar um arquivo de configuração para modificar as restrições de contexto de segurança (SCCs) do OpenShift para contêineres Trident Protect após instalar o Trident Protect. Essas restrições definem as limitações de segurança para os pods em um cluster Red Hat OpenShift.

Passos

1. Crie um arquivo chamado `sccconfig.yaml`.
2. Adicione a opção SCC ao arquivo e modifique os parâmetros de acordo com as necessidades do seu ambiente.

   O exemplo a seguir mostra os valores padrão dos parâmetros para a opção SCC:

```
scc:
  create: true
  name: trident-protect-job
  priority: 1
```

Esta tabela descreve os parâmetros para a opção SCC:

Parâmetro	Descrição	Padrão
criar	Determina se um recurso SCC pode ser criado. Um recurso SCC será criado somente se `scc.create` estiver definido como `true` e o processo de instalação Helm identificar um ambiente OpenShift. Se não estiver operando no OpenShift, ou se `scc.create` estiver definido como `false`, nenhum recurso SCC será criado.	verdadeiro
nome	Especifica o nome do SCC.	Trident-protect-job
prioridade	Define a prioridade do SCC. Os SCCs com valores de prioridade mais elevados são avaliados antes daqueles com valores mais baixos.	1

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

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

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

Configure as definições adicionais do gráfico de navegação do Trident Protect.

Você pode personalizar as configurações do AutoSupport e a filtragem de namespace para atender aos seus requisitos específicos. A tabela a seguir descreve os parâmetros de configuração disponíveis:

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

Parâmetro	Tipo	Descrição
autoSupport.inseguro	booleano	Ignora a verificação TLS para conexões proxy AutoSupport quando definido como `true` . Use somente para conexões proxy inseguras. (padrão: `false`)
autoSupport.habilitado	booleano	Ativa ou desativa os uploads diários do pacote Trident Protect AutoSupport . Quando definido para `false` Os uploads diários agendados estão desativados, mas você ainda pode gerar pacotes de suporte manualmente. (padrão: `true`)
restaurarSkipNamespaceAnnotations	cadeia de carateres	Lista separada por vírgulas de anotações de namespace a serem excluídas das operações de backup e restauração. Permite filtrar namespaces com base em anotações.
restaurarIgnorarEtiquetasDeEspaçoDeNomes	cadeia de carateres	Lista separada por vírgulas de rótulos de namespace a serem excluídos das operações de backup e restauração. Permite filtrar namespaces com base em rótulos.

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

Usar arquivo YAML

Passos

1. Crie um arquivo de configuração e nomeie-o `values.yaml` .
2. No arquivo que você criou, adicione as opções de configuração que deseja personalizar.

```
autoSupport:
  enabled: false
  proxy: http://my.proxy.url
  insecure: true
restoreSkipNamespaceAnnotations: "annotation1,annotation2"
restoreSkipNamespaceLabels: "label1,label2"
```

3. Depois de preencher o `values.yaml` arquivo com os valores corretos, aplique o arquivo de configuração:

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

Usar sinalizador CLI

Passos

1. Use o seguinte comando com o `--set` sinalizador para especificar parâmetros individuais:

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect \
  --set autoSupport.enabled=false \
  --set autoSupport.proxy=http://my.proxy.url \
  --set-string
restoreSkipNamespaceAnnotations="{annotation1,annotation2}" \
  --set-string restoreSkipNamespaceLabels="{label1,label2}" \
  --reuse-values
```

Restrinja os pods do Trident Protect a nós específicos.

Você pode usar a restrição de seleção de nós `nodeSelector` do Kubernetes para controlar quais de seus nós são elegíveis para executar pods do Trident Protect, com base nos rótulos dos nós. Por padrão, o Trident Protect é restrito a nós que executam Linux. Você pode personalizar ainda mais essas restrições de acordo com suas necessidades.

Passos

1. Crie um arquivo chamado `nodeSelectorConfig.yaml`.
2. Adicione a opção nodeSeletor ao arquivo e modifique o arquivo para adicionar ou alterar rótulos de nó

para restringir de acordo com as necessidades do seu ambiente. Por exemplo, o arquivo a seguir contém a restrição padrão do sistema operacional, mas também tem como alvo uma região específica e nome do aplicativo:

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

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

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

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

Gerenciar Trident Protect

Gerencie a autorização e o controle de acesso do Trident Protect.

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

O administrador do cluster sempre tem acesso a recursos no namespace padrão `trident-protect` e também pode acessar recursos em todos os outros namespaces. Para controlar o acesso a recursos e aplicações, é necessário criar espaços de nomes adicionais e adicionar recursos e aplicações a esses espaços de nomes.

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

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

- **AppVault**: Requer dados de credenciais de bucket
- **Pacote de Suporte Automático**: Coleta métricas, registros e outros dados confidenciais do Trident Protect.
- **AutoSupportBundleSchedule**: Gerencia os horários de coleta de Registros

Como prática recomendada, use o RBAC para restringir o acesso a objetos privilegiados aos administradores.

Para obter mais informações sobre como o RBAC regula o acesso a recursos e namespaces, consulte o "Documentação do Kubernetes RBAC".

Para obter informações sobre contas de serviço, consulte o "Documentação da conta de serviço do Kubernetes".

Exemplo: Gerencie o acesso para dois grupos de usuários

Por exemplo, uma organização tem um administrador de cluster, um grupo de usuários de engenharia e um grupo de usuários de marketing. O administrador do cluster concluiria as seguintes tarefas para criar um ambiente onde o grupo de engenharia e o grupo de marketing tenham acesso apenas aos recursos atribuídos aos respetivos namespaces.

Etapa 1: Crie um namespace para conter recursos para cada grupo

Criar um namespace permite separar recursos logicamente e controlar melhor quem tem acesso a esses recursos.

Passos

1. Crie um namespace para o grupo de engenharia:

```
kubectl create ns engineering-ns
```

2. Crie um namespace para o grupo de marketing:

```
kubectl create ns marketing-ns
```

Etapa 2: Crie novas contas de serviço para interagir com recursos em cada namespace

Cada novo namespace que você criar vem com uma conta de serviço padrão, mas você deve criar uma conta de serviço para cada grupo de usuários para que você possa dividir ainda mais Privileges entre grupos no futuro, se necessário.

Passos

1. Crie uma conta de serviço para o grupo de engenharia:

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: eng-user
  namespace: engineering-ns
```

2. Crie uma conta de serviço para o grupo de marketing:

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: mkt-user
  namespace: marketing-ns
```

Passo 3: Crie um segredo para cada nova conta de serviço

Um segredo de conta de serviço é usado para autenticar com a conta de serviço e pode ser facilmente excluído e recriado se comprometido.

Passos

1. Crie um segredo para a conta de serviço de engenharia:

```
apiVersion: v1
kind: Secret
metadata:
  annotations:
    kubernetes.io/service-account.name: eng-user
  name: eng-user-secret
  namespace: engineering-ns
type: kubernetes.io/service-account-token
```

2. Crie um segredo para a conta do serviço de marketing:

```
apiVersion: v1
kind: Secret
metadata:
  annotations:
    kubernetes.io/service-account.name: mkt-user
  name: mkt-user-secret
  namespace: marketing-ns
type: kubernetes.io/service-account-token
```

Passo 4: Crie um objeto RoleBinding para vincular o objeto ClusterRole a cada nova conta de serviço

Um objeto ClusterRole padrão é criado quando você instala o Trident Protect. Você pode vincular essa ClusterRole à conta de serviço criando e aplicando um objeto RoleBinding.

Passos

1. Vincule o ClusterRole à conta de serviço de engenharia:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: engineering-ns-tenant-rolebinding
  namespace: engineering-ns
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: trident-protect-tenant-cluster-role
subjects:
- kind: ServiceAccount
  name: eng-user
  namespace: engineering-ns
```

2. Vincule o ClusterRole à conta do serviço de marketing:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: marketing-ns-tenant-rolebinding
  namespace: marketing-ns
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: trident-protect-tenant-cluster-role
subjects:
- kind: ServiceAccount
  name: mkt-user
  namespace: marketing-ns
```

Passo 5: Testar permissões

Teste se as permissões estão corretas.

Passos

1. Confirme se os usuários de engenharia podem acessar os recursos de engenharia:

```
kubectl auth can-i --as=system:serviceaccount:engineering-ns:eng-user
get applications.protect.trident.netapp.io -n engineering-ns
```

2. Confirme que os usuários de engenharia não podem acessar recursos de marketing:

```
kubectl auth can-i --as=system:serviceaccount:engineering-ns:eng-user
get applications.protect.trident.netapp.io -n marketing-ns
```

Etapa 6: Conceder acesso a objetos AppVault

Para executar tarefas de gerenciamento de dados, como backups e snapshots, o administrador do cluster precisa conceder acesso a objetos AppVault a usuários individuais.

Passos

1. Crie e aplique um arquivo YAML de combinação secreta e AppVault que concede a um usuário acesso a um AppVault. Por exemplo, o CR a seguir concede acesso a um AppVault ao usuário `eng-user`:

```
apiVersion: v1
data:
  accessKeyID: <ID_value>
  secretAccessKey: <key_value>
kind: Secret
metadata:
  name: appvault-for-eng-user-only-secret
  namespace: trident-protect
type: Opaque
---
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: appvault-for-eng-user-only
  namespace: trident-protect # Trident Protect system namespace
spec:
  providerConfig:
    azure:
      accountName: ""
      bucketName: ""
      endpoint: ""
    gcp:
      bucketName: ""
      projectID: ""
    s3:
      bucketName: testbucket
      endpoint: 192.168.0.1:30000
      secure: "false"
      skipCertValidation: "true"
  providerCredentials:
    accessKeyID:
      valueFromSecret:
        key: accessKeyID
        name: appvault-for-eng-user-only-secret
    secretAccessKey:
      valueFromSecret:
        key: secretAccessKey
        name: appvault-for-eng-user-only-secret
  providerType: GenericS3
```

2. Crie e aplique um CR de função para permitir que os administradores de cluster concedam acesso a recursos específicos em um namespace. Por exemplo:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: eng-user-appvault-reader
  namespace: trident-protect
rules:
- apiGroups:
  - protect.trident.netapp.io
  resourceNames:
  - appvault-for-enguser-only
  resources:
  - appvaults
  verbs:
  - get
```

3. Criar e aplicar um RoleBinding CR para vincular as permissões ao usuário eng-user. Por exemplo:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: eng-user-read-appvault-binding
  namespace: trident-protect
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: eng-user-appvault-reader
subjects:
- kind: ServiceAccount
  name: eng-user
  namespace: engineering-ns
```

4. Verifique se as permissões estão corretas.

   a. Tente recuperar informações de objeto AppVault para todos os namespaces:

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

   Você deve ver saída semelhante ao seguinte:

```
Error from server (Forbidden): appvaults.protect.trident.netapp.io is
forbidden: User "system:serviceaccount:engineering-ns:eng-user"
cannot list resource "appvaults" in API group
"protect.trident.netapp.io" in the namespace "trident-protect"
```

b. Teste para ver se o usuário pode obter as informações do AppVault que ele agora tem permissão para acessar:

```
kubectl auth can-i --as=system:serviceaccount:engineering-ns:eng-user
get appvaults.protect.trident.netapp.io/appvault-for-eng-user-only -n
trident-protect
```

Você deve ver saída semelhante ao seguinte:

```
yes
```

Resultado

Os usuários aos quais você concedeu permissões AppVault devem poder usar objetos AppVault autorizados para operações de gerenciamento de dados de aplicativos e não devem poder acessar recursos fora dos namespaces atribuídos ou criar novos recursos aos quais eles não têm acesso.

Monitorar recursos do Trident Protect

Você pode usar as ferramentas de código aberto kube-state-metrics, Prometheus e Alertmanager para monitorar a integridade dos recursos protegidos pelo Trident Protect.

O serviço kube-state-metrics gera métricas a partir da comunicação da API do Kubernetes. Ao utilizá-lo com o Trident Protect, você obtém informações úteis sobre o estado dos recursos em seu ambiente.

O Prometheus é um conjunto de ferramentas que pode ingerir os dados gerados pelo kube-state-metrics e apresentá-los como informações facilmente legíveis sobre esses objetos. Em conjunto, o kube-state-metrics e o Prometheus oferecem uma maneira de monitorar a integridade e o status dos recursos que você está gerenciando com o Trident Protect.

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

> As configurações e orientações incluídas nessas etapas são apenas exemplos; você precisa personalizá-las para corresponder ao seu ambiente. Consulte a seguinte documentação oficial para obter instruções e suporte específicos:
>
> - "documentação de métricas de estado do kube"
> - "Documentação do Prometheus"
> - "Documentação do Alertmanager"

Passo 1: Instale as ferramentas de monitoramento

Para habilitar o monitoramento de recursos no Trident Protect, você precisa instalar e configurar o kube-state-metrics, o Prometheus e o Alertmanager.

Instalar métricas de estado do kube

Você pode instalar métricas de estado do kube usando o Helm.

Passos

1. Adicione o gráfico Helm de métricas de estado kube. Por exemplo:

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

2. Aplique o Prometheus ServiceMonitor CRD ao cluster:

```
kubectl apply -f https://raw.githubusercontent.com/prometheus-
operator/prometheus-operator/main/example/prometheus-operator-
crd/monitoring.coreos.com_servicemonitors.yaml
```

3. Crie um arquivo de configuração para o gráfico Helm (por exemplo, `metrics-config.yaml`). Você pode personalizar o seguinte exemplo de configuração para corresponder ao seu ambiente:

Metrics-config.yaml: Configuração do gráfico Helm do kube-State-metrics

```
---
extraArgs:
  # Collect only custom metrics
  - --custom-resource-state-only=true

customResourceState:
  enabled: true
  config:
    kind: CustomResourceStateMetrics
    spec:
      resources:
      - groupVersionKind:
          group: protect.trident.netapp.io
          kind: "Backup"
          version: "v1"
        labelsFromPath:
          backup_uid: [metadata, uid]
          backup_name: [metadata, name]
          creation_time: [metadata, creationTimestamp]
        metrics:
        - name: backup_info
          help: "Exposes details about the Backup state"
          each:
            type: Info
            info:
              labelsFromPath:
                appVaultReference: ["spec", "appVaultRef"]
                appReference: ["spec", "applicationRef"]
rbac:
  extraRules:
  - apiGroups: ["protect.trident.netapp.io"]
    resources: ["backups"]
    verbs: ["list", "watch"]

# Collect metrics from all namespaces
namespaces: ""

# Ensure that the metrics are collected by Prometheus
prometheus:
  monitor:
    enabled: true
```

4. Instale as métricas de estado do kube implantando o gráfico Helm. Por exemplo:

```
helm install custom-resource -f metrics-config.yaml prometheus-
community/kube-state-metrics --version 5.21.0
```

5. Configure o kube-state-metrics para gerar métricas para os recursos personalizados usados pelo Trident Protect, seguindo as instruções em "documentação de recursos personalizados de métricas de estado do kube" .

Instale Prometheus

Você pode instalar o Prometheus seguindo as instruções no "Documentação do Prometheus".

Instale o Alertmanager

Você pode instalar o Alertmanager seguindo as instruções no "Documentação do Alertmanager".

Passo 2: Configure as ferramentas de monitoramento para trabalhar em conjunto

Depois de instalar as ferramentas de monitoramento, você precisa configurá-las para trabalhar em conjunto.

Passos

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

   prometheus.yaml: integração do serviço kube-state-metrics com o Prometheus

```
---
apiVersion: v1
kind: ConfigMap
metadata:
  name: prometheus-config
  namespace: trident-protect
data:
  prometheus.yaml: |
    global:
      scrape_interval: 15s
    scrape_configs:
      - job_name: 'kube-state-metrics'
        static_configs:
          - targets: ['kube-state-metrics.trident-protect.svc:8080']
```

2. Configure Prometheus para rotear alertas para Alertmanager. Edite o arquivo de configuração Prometheus (`prometheus.yaml`) e adicione a seguinte seção:

prometheus.yaml: Enviar alertas para o Alertmanager

```
alerting:
  alertmanagers:
    - static_configs:
        - targets:
            - alertmanager.trident-protect.svc:9093
```

Resultado

Prometheus agora pode coletar métricas de kube-State-metrics e enviar alertas para Alertmanager. Agora você está pronto para configurar quais condições acionam um alerta e onde os alertas devem ser enviados.

Etapa 3: Configurar alertas e destinos de alerta

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

Exemplo de alerta: Falha de backup

O exemplo a seguir define um alerta crítico que é acionado quando o status do recurso personalizado de backup é definido como `Error` por 5 segundos ou mais. Você pode personalizar este exemplo para corresponder ao seu ambiente e incluir esse snippet YAML em seu `prometheus.yaml` arquivo de configuração:

rules.yaml: Defina um alerta do Prometheus para backups com falha

```
rules.yaml: |
  groups:
    - name: fail-backup
        rules:
          - alert: BackupFailed
            expr: kube_customresource_backup_info{status="Error"}
            for: 5s
            labels:
              severity: critical
            annotations:
              summary: "Backup failed"
              description: "A backup has failed."
```

Configure o Alertmanager para enviar alertas para outros canais

Você pode configurar o Alertmanager para enviar notificações para outros canais, como e-mail, PagerDuty, Microsoft Teams ou outros serviços de notificação especificando a respetiva configuração no `alertmanager.yaml` arquivo.

O exemplo a seguir configura o Alertmanager para enviar notificações para um canal do Slack. Para personalizar este exemplo para o ambiente, substitua o valor da `api_url` chave pelo URL do webhook do Slack usado no ambiente:

alertmanager.yaml: Enviar alertas para um canal do Slack

```
data:
  alertmanager.yaml: |
    global:
      resolve_timeout: 5m
    route:
      receiver: 'slack-notifications'
    receivers:
      - name: 'slack-notifications'
        slack_configs:
          - api_url: '<your-slack-webhook-url>'
            channel: '#failed-backups-channel'
            send_resolved: false
```

Gere um pacote de suporte Trident Protect

O Trident Protect permite que os administradores gerem pacotes que incluem informações úteis para o Suporte da NetApp , como logs, métricas e informações de topologia sobre os clusters e aplicativos gerenciados. Se você estiver conectado à Internet, poderá carregar pacotes de suporte no NetApp Support Site (NSS) usando um arquivo de recurso personalizado (CR).

Crie um pacote de suporte usando um CR

Passos

1. Crie o arquivo de recurso personalizado (CR) e nomeie-o (por exemplo, `trident-protect-support-bundle.yaml`).
2. Configure os seguintes atributos:
   - **metadata.name**: (*required*) o nome deste recurso personalizado; escolha um nome único e sensível para o seu ambiente.
   - **Spec.triggerType**: (*required*) determina se o pacote de suporte é gerado imediatamente ou programado. A geração de pacotes programados acontece às 12AM UTC. Valores possíveis:
     - Programado
     - Manual
   - **Spec.uploadEnabled**: (*Optional*) controla se o pacote de suporte deve ser carregado para o site de suporte da NetApp depois que ele é gerado. Se não for especificado, o padrão é `false`. Valores possíveis:
     - verdadeiro
     - falso (padrão)
   - **Spec.dataWindowStart**: (*Optional*) Uma cadeia de carateres de data no formato RFC 3339 que especifica a data e a hora em que a janela de dados incluídos no pacote de suporte deve começar. Se não for especificado, o padrão é 24 horas atrás. A data da janela mais antiga que você pode especificar é de 7 dias atrás.

     Exemplo YAML:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: AutoSupportBundle
metadata:
  name: trident-protect-support-bundle
spec:
  triggerType: Manual
  uploadEnabled: true
  dataWindowStart: 2024-05-05T12:30:00Z
```

3. Depois de preencher o `trident-protect-support-bundle.yaml` arquivo com os valores corretos, aplique o CR:

```
kubectl apply -f trident-protect-support-bundle.yaml -n trident-protect
```

Crie um pacote de suporte usando a CLI

Passos

1. Crie o pacote de suporte, substituindo valores entre parênteses por informações do seu ambiente. O

`trigger-type` determina se o pacote é criado imediatamente ou se o tempo de criação é ditado pelo agendamento e pode ser `Manual` ou `Scheduled`. A predefinição é `Manual`.

Por exemplo:

```
tridentctl-protect create autosupportbundle <my-bundle-name>
--trigger-type <trigger-type> -n trident-protect
```

Monitore e recupere o pacote de suporte

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

Passos

1. Espere pelo `status.generationState` alcançar `Completed` estado. Você pode monitorar o progresso da geração com o seguinte comando:

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

2. Recupere o pacote de suporte para seu sistema local. Obtenha o comando de cópia do pacote AutoSupport concluído:

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

   Encontre o `kubectl cp` comando da saída e execute-o, substituindo o argumento de destino pelo seu diretório local preferido.

Aprimore o Trident Protect

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

- Ao atualizar a partir da versão 24.10, os snapshots executados durante a atualização podem falhar. Essa falha não impede a criação de snapshots futuros, sejam eles manuais ou agendados. Se um snapshot falhar durante a atualização, você pode criar um novo snapshot manualmente para garantir a proteção do seu aplicativo.

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

- Para instalações em registros privados, certifique-se de que o gráfico Helm e as imagens necessárias para a versão desejada estejam disponíveis em seu registro privado e verifique se seus valores Helm personalizados são compatíveis com a nova versão do gráfico. Para obter mais informações, consulte"Instale o Trident Protect a partir de um registro privado." .

Para atualizar o Trident Protect, siga os passos abaixo.

Passos

1. Atualize o repositório Helm do Trident:

```
helm repo update
```

2. Atualize os CRDs do Trident Protect:

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

   a. Execute este comando para mudar o gerenciamento de CRDs de `trident-protect-crds` para `trident-protect` :

```
kubectl get crd | grep protect.trident.netapp.io | awk '{print $1}' |
xargs -I {} kubectl patch crd {} --type merge -p '{"metadata":
{"annotations":{"meta.helm.sh/release-name": "trident-protect"}}}'
```

   b. Execute este comando para excluir o segredo do Helm para o `trident-protect-crds` gráfico:

   Não desinstale o `trident-protect-crds` gráfico usando o Helm, pois isso pode remover seus CRDs e quaisquer dados relacionados.

```
kubectl delete secret -n trident-protect -l name=trident-protect-
crds,owner=helm
```

3. Aprimore o Trident Protect:

```
helm upgrade trident-protect netapp-trident-protect/trident-protect
--version 100.2510.0 --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 a resolução de problemas, pois ajuda o suporte da NetApp a diagnosticar problemas sem exigir alterações no nível de registro ou reprodução do problema.

Gerenciar e proteger aplicativos

Use objetos do Trident Protect AppVault para gerenciar buckets.

O recurso personalizado (CR) do bucket para o Trident Protect é conhecido como AppVault. Os objetos do AppVault são a representação declarativa do fluxo de trabalho do Kubernetes de um bucket de armazenamento. Um AppVault CR contém as configurações necessárias para que um bucket seja usado em operações de proteção, como backups, snapshots, operações de restauração e replicação do SnapMirror . Somente administradores podem criar AppVaults.

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

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

Configurar a autenticação e as senhas do AppVault

Antes de criar um AppVault CR, certifique-se de que o AppVault e o movimentador de dados escolhido possam ser autenticados com o provedor e quaisquer recursos relacionados.

Senhas do repositório do controlador de dados

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

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

Crie um segredo de senha do repositório do mover de dados

Utilize os exemplos a seguir para criar a senha secreta. Ao criar objetos do AppVault, você pode instruir o Trident Protect a usar esse segredo para autenticar com o repositório de movimentação de dados.

- Dependendo do motor de dados que você está usando, você só precisa incluir a senha correspondente para esse controlador de dados. Por exemplo, se você estiver usando Restic e não planeja usar o Kopia no futuro, você pode incluir apenas a senha Restic quando criar o segredo.
- Guarde a senha em um local seguro. Você precisará dela para restaurar dados no mesmo cluster ou em um diferente. Se o cluster ou o `trident-protect` namespace for excluído, você não poderá restaurar seus backups ou snapshots sem a senha.

Use um CR

```
---
apiVersion: v1
data:
  KOPIA_PASSWORD: <base64-encoded-password>
  RESTIC_PASSWORD: <base64-encoded-password>
kind: Secret
metadata:
  name: my-optional-data-mover-secret
  namespace: trident-protect
type: Opaque
```

Use a CLI

```
kubectl create secret generic my-optional-data-mover-secret \
--from-literal=KOPIA_PASSWORD=<plain-text-password> \
--from-literal=RESTIC_PASSWORD=<plain-text-password> \
-n trident-protect
```

Permissões de IAM de armazenamento compatível com S3

Ao acessar armazenamento compatível com S3, como Amazon S3, S3 genérico, "StorageGRID S3" , ou "ONTAP S3" Ao usar o Trident Protect, você precisa garantir que as credenciais de usuário fornecidas tenham as permissões necessárias para acessar o bucket. Segue abaixo um exemplo de uma política que concede as permissões mínimas necessárias para acesso ao Trident Protect. Você pode aplicar essa política ao usuário que gerencia as políticas de bucket compatíveis com o S3.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:GetObject",
        "s3:ListBucket",
        "s3:DeleteObject"
      ],
      "Resource": "*"
    }
  ]
}
```

Para obter mais informações sobre as políticas do Amazon S3, consulte os exemplos no "Documentação do Amazon S3" .

Identidade do Pod EKS para autenticação do Amazon S3 (AWS)

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

Requisitos para a identidade do EKS Pod com o Trident Protect

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

- Seu cluster EKS tem a Identidade do Pod habilitada.
- Você criou uma função do IAM com as permissões de bucket do S3 necessárias. Para saber mais, consulte"Permissões de IAM de armazenamento compatível com S3" .
- A função IAM está associada às seguintes contas de serviço do Trident Protect:
  - `<trident-protect>-controller-manager`
  - `<trident-protect>-resource-backup`
  - `<trident-protect>-resource-restore`
  - `<trident-protect>-resource-delete`

Para obter instruções detalhadas sobre como habilitar a identidade do pod e associar funções do IAM a contas de serviço, consulte o "Documentação de identidade do pod AWS EKS" .

Configuração do AppVault Ao usar o EKS Pod Identity, configure seu AppVault CR com o `useIAM: true` sinalizador em vez de credenciais explícitas:

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: eks-protect-vault
  namespace: trident-protect
spec:
  providerType: AWS
  providerConfig:
    s3:
      bucketName: trident-protect-aws
      endpoint: s3.example.com
      useIAM: true
```

Exemplos de geração de chaves AppVault para provedores de nuvem

Ao definir um AppVault CR, você precisa incluir credenciais para acessar os recursos hospedados pelo provedor, a menos que esteja usando a autenticação do IAM. A maneira como você gera as chaves para as credenciais varia de acordo com o provedor. A seguir estão exemplos de geração de chaves de linha de comando para vários provedores. Você pode usar os exemplos a seguir para criar chaves para as credenciais de cada provedor de nuvem.

Google Cloud

```
kubectl create secret generic <secret-name> \
--from-file=credentials=<mycreds-file.json> \
-n trident-protect
```

Amazon S3 (AWS)

```
kubectl create secret generic <secret-name> \
--from-literal=accessKeyID=<objectstorage-accesskey> \
--from-literal=secretAccessKey=<amazon-s3-trident-protect-src-bucket
-secret> \
-n trident-protect
```

Microsoft Azure

```
kubectl create secret generic <secret-name> \
--from-literal=accountKey=<secret-name> \
-n trident-protect
```

Genérico S3

```
kubectl create secret generic <secret-name> \
--from-literal=accessKeyID=<objectstorage-accesskey> \
--from-literal=secretAccessKey=<generic-s3-trident-protect-src-bucket
-secret> \
-n trident-protect
```

ONTAP S3

```
kubectl create secret generic <secret-name> \
--from-literal=accessKeyID=<objectstorage-accesskey> \
--from-literal=secretAccessKey=<ontap-s3-trident-protect-src-bucket
-secret> \
-n trident-protect
```

StorageGRID S3

```
kubectl create secret generic <secret-name> \
--from-literal=accessKeyID=<objectstorage-accesskey> \
--from-literal=secretAccessKey=<storagegrid-s3-trident-protect-src
-bucket-secret> \
-n trident-protect
```

Exemplos de criação do AppVault

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

Exemplos do AppVault CR

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

- Opcionalmente, você pode especificar um segredo do Kubernetes que contém senhas personalizadas para a criptografia do repositório Restic e Kopia. Senhas do repositório do controlador de dadosConsulte para obter mais informações.
- Para objetos do Amazon S3 (AWS) AppVault, você pode especificar opcionalmente um sessionToken, o que é útil se você estiver usando SSO (logon único) para autenticação. Esse token é criado quando você gera chaves para o provedor no Exemplos de geração de chaves AppVault para provedores de nuvem.
- Para objetos S3 AppVault, você pode opcionalmente especificar um URL de proxy de saída para tráfego S3 de saída usando a `spec.providerConfig.S3.proxyURL` chave.

Google Cloud

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: gcp-trident-protect-src-bucket
  namespace: trident-protect
spec:
  dataMoverPasswordSecretRef: my-optional-data-mover-secret
  providerType: GCP
  providerConfig:
    gcp:
      bucketName: trident-protect-src-bucket
      projectID: project-id
  providerCredentials:
    credentials:
      valueFromSecret:
        key: credentials
        name: gcp-trident-protect-src-bucket-secret
```

Amazon S3 (AWS)

```
---
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: amazon-s3-trident-protect-src-bucket
  namespace: trident-protect
spec:
  dataMoverPasswordSecretRef: my-optional-data-mover-secret
  providerType: AWS
  providerConfig:
    s3:
      bucketName: trident-protect-src-bucket
      endpoint: s3.example.com
      proxyURL: http://10.1.1.1:3128
  providerCredentials:
    accessKeyID:
      valueFromSecret:
        key: accessKeyID
        name: s3-secret
    secretAccessKey:
      valueFromSecret:
        key: secretAccessKey
        name: s3-secret
    sessionToken:
      valueFromSecret:
        key: sessionToken
        name: s3-secret
```

Para ambientes EKS usando Pod Identity com o movimentador de dados Kopia, você pode remover o `providerCredentials` seção e adicionar `useIAM: true` sob o `s3` configuração em vez disso.

Microsoft Azure

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: azure-trident-protect-src-bucket
  namespace: trident-protect
spec:
  dataMoverPasswordSecretRef: my-optional-data-mover-secret
  providerType: Azure
  providerConfig:
    azure:
      accountName: account-name
      bucketName: trident-protect-src-bucket
  providerCredentials:
    accountKey:
      valueFromSecret:
        key: accountKey
        name: azure-trident-protect-src-bucket-secret
```

Genérico S3

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: generic-s3-trident-protect-src-bucket
  namespace: trident-protect
spec:
  dataMoverPasswordSecretRef: my-optional-data-mover-secret
  providerType: GenericS3
  providerConfig:
    s3:
      bucketName: trident-protect-src-bucket
      endpoint: s3.example.com
      proxyURL: http://10.1.1.1:3128
  providerCredentials:
    accessKeyID:
      valueFromSecret:
        key: accessKeyID
        name: s3-secret
    secretAccessKey:
      valueFromSecret:
        key: secretAccessKey
        name: s3-secret
```

ONTAP S3

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: ontap-s3-trident-protect-src-bucket
  namespace: trident-protect
spec:
  dataMoverPasswordSecretRef: my-optional-data-mover-secret
  providerType: OntapS3
  providerConfig:
    s3:
      bucketName: trident-protect-src-bucket
      endpoint: s3.example.com
      proxyURL: http://10.1.1.1:3128
  providerCredentials:
    accessKeyID:
      valueFromSecret:
        key: accessKeyID
        name: s3-secret
    secretAccessKey:
      valueFromSecret:
        key: secretAccessKey
        name: s3-secret
```

StorageGRID S3

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: storagegrid-s3-trident-protect-src-bucket
  namespace: trident-protect
spec:
  dataMoverPasswordSecretRef: my-optional-data-mover-secret
  providerType: StorageGridS3
  providerConfig:
    s3:
      bucketName: trident-protect-src-bucket
      endpoint: s3.example.com
      proxyURL: http://10.1.1.1:3128
  providerCredentials:
    accessKeyID:
      valueFromSecret:
        key: accessKeyID
        name: s3-secret
    secretAccessKey:
      valueFromSecret:
        key: secretAccessKey
        name: s3-secret
```

Exemplos de criação do AppVault usando a CLI do Trident Protect

Você pode usar os seguintes exemplos de comandos CLI para criar o AppVault CRS para cada provedor.

- Opcionalmente, você pode especificar um segredo do Kubernetes que contém senhas personalizadas para a criptografia do repositório Restic e Kopia. Senhas do repositório do controlador de dadosConsulte para obter mais informações.
- Para objetos S3 AppVault, você pode opcionalmente especificar um URL de proxy de saída para tráfego S3 de saída usando o `--proxy-url <ip_address:port>` argumento.

Google Cloud

```
tridentctl-protect create vault GCP <vault-name> \
--bucket <mybucket> \
--project <my-gcp-project> \
--secret <secret-name>/credentials \
--data-mover-password-secret-ref <my-optional-data-mover-secret> \
-n trident-protect
```

Amazon S3 (AWS)

```
tridentctl-protect create vault AWS <vault-name> \
--bucket <bucket-name> \
--secret  <secret-name>  \
--endpoint <s3-endpoint> \
--data-mover-password-secret-ref <my-optional-data-mover-secret> \
-n trident-protect
```

Microsoft Azure

```
tridentctl-protect create vault Azure <vault-name> \
--account <account-name> \
--bucket <bucket-name> \
--secret <secret-name> \
--data-mover-password-secret-ref <my-optional-data-mover-secret> \
-n trident-protect
```

Genérico S3

```
tridentctl-protect create vault GenericS3 <vault-name> \
--bucket <bucket-name> \
--secret  <secret-name>  \
--endpoint <s3-endpoint> \
--data-mover-password-secret-ref <my-optional-data-mover-secret> \
-n trident-protect
```

ONTAP S3

```
tridentctl-protect create vault OntapS3 <vault-name> \
--bucket <bucket-name> \
--secret  <secret-name>  \
--endpoint <s3-endpoint> \
--data-mover-password-secret-ref <my-optional-data-mover-secret> \
-n trident-protect
```

StorageGRID S3

```
tridentctl-protect create vault StorageGridS3 <vault-name> \
--bucket <bucket-name> \
--secret  <secret-name>  \
--endpoint <s3-endpoint> \
--data-mover-password-secret-ref <my-optional-data-mover-secret> \
-n trident-protect
```

Apoiado `providerConfig.s3` opções de configuração

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

Parâmetro	Descrição	Padrão	Exemplo
`providerConfig.s3.skipCertValidation`	Desative a verificação de certificado SSL/TLS.	falso	"verdadeiro", "falso"
`providerConfig.s3.secure`	Habilite a comunicação HTTPS segura com o endpoint S3.	verdadeiro	"verdadeiro", "falso"
`providerConfig.s3.proxyURL`	Especifique o URL do servidor proxy usado para conectar-se ao S3.	Nenhum	http://proxy.example.com:8080
`providerConfig.s3.rootCA`	Forneça um certificado CA raiz personalizado para verificação SSL/TLS.	Nenhum	"CN=MyCustomCA"
`providerConfig.s3.useIAM`	Habilite a autenticação IAM para acessar os buckets do S3. Aplicável à identidade do Pod EKS.	falso	verdadeiro, falso

Ver informações do AppVault

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

Passos

1. Exibir o conteúdo de um objeto AppVault:

```
tridentctl-protect get appvaultcontent gcp-vault \
--show-resources all \
-n trident-protect
```

Exemplo de saída:

```
+-------------+-------+----------+----------------------------
+---------------------------+
|   CLUSTER   |  APP  |   TYPE   |            NAME             |
TIMESTAMP         |
+-------------+-------+----------+----------------------------
+---------------------------+
|             | mysql | snapshot | mysnap                      | 2024-
08-09 21:02:11 (UTC) |
| production1 | mysql | snapshot | hourly-e7db6-20240815180300 | 2024-
08-15 18:03:06 (UTC) |
| production1 | mysql | snapshot | hourly-e7db6-20240815190300 | 2024-
08-15 19:03:06 (UTC) |
| production1 | mysql | snapshot | hourly-e7db6-20240815200300 | 2024-
08-15 20:03:06 (UTC) |
| production1 | mysql | backup   | hourly-e7db6-20240815180300 | 2024-
08-15 18:04:25 (UTC) |
| production1 | mysql | backup   | hourly-e7db6-20240815190300 | 2024-
08-15 19:03:30 (UTC) |
| production1 | mysql | backup   | hourly-e7db6-20240815200300 | 2024-
08-15 20:04:21 (UTC) |
| production1 | mysql | backup   | mybackup5                   | 2024-
08-09 22:25:13 (UTC) |
|             | mysql | backup   | mybackup                    | 2024-
08-09 21:02:52 (UTC) |
+-------------+-------+----------+----------------------------
+---------------------------+
```

2. Opcionalmente, para ver o AppVaultPath para cada recurso, use o `--show-paths` sinalizador .

   O nome do cluster na primeira coluna da tabela só estará disponível se um nome de cluster tiver sido especificado na instalação do Trident Protect no Helm. Por exemplo: `--set clusterName=production1` .

Remova um AppVault

Você pode remover um objeto AppVault a qualquer momento.

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

Antes de começar

Certifique-se de que você excluiu todos os CRS de snapshot e backup que estão sendo usados pelo AppVault que deseja excluir.

Remova um AppVault usando a CLI do Kubernetes

1. Remova o objeto AppVault, substituindo `appvault-name` pelo nome do objeto AppVault para remover:

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

Remova um AppVault usando a CLI do Trident Protect.

1. Remova o objeto AppVault, substituindo `appvault-name` pelo nome do objeto AppVault para remover:

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

Defina uma aplicação para gestão com o Trident Protect.

Você pode definir um aplicativo que deseja gerenciar com o Trident Protect criando um CR (Create Request) de aplicativo e um CR do AppVault associado.

Crie um AppVault CR

Você precisa criar um AppVault CR que será usado ao executar operações de proteção de dados no aplicativo, e o AppVault CR precisa residir no cluster onde o Trident Protect está instalado. O pedido de configuração (CR) do AppVault é específico para o seu ambiente; para exemplos de pedidos de configuração do AppVault, consulte [link para a documentação]."Recursos personalizados do AppVault."

Definir uma aplicação

Você precisa definir cada aplicativo que deseja gerenciar com o Trident Protect. Você pode definir um aplicativo para gerenciamento criando manualmente uma solicitação de configuração (CR) do aplicativo ou usando a CLI do Trident Protect.

Adicione uma aplicação utilizando um CR

Passos

1. Criar o ficheiro CR da aplicação de destino:
   a. Crie o arquivo de recurso personalizado (CR) e nomeie-o (por exemplo, `maria-app.yaml`).
   b. Configure os seguintes atributos:
      - **metadata.name**: (*required*) o nome do recurso personalizado do aplicativo. Observe o nome escolhido porque outros arquivos CR necessários para operações de proteção referem-se a esse valor.
      - **spec.includedNamespaces**: (*required*) Use o seletor de namespace e rótulo para especificar os namespaces e recursos que o aplicativo usa. O namespace do aplicativo deve fazer parte dessa lista. O seletor de etiquetas é opcional e pode ser usado para filtrar recursos dentro de cada namespace especificado.
      - **spec.includedClusterScopedResources**: (*Optional*) Use este atributo para especificar recursos com escopo de cluster a serem incluídos na definição do aplicativo. Esse atributo permite que você selecione esses recursos com base em seu grupo, versão, tipo e rótulos.
        - **GroupVersionKind**: (*required*) especifica o grupo API, a versão e o tipo do recurso com escopo de cluster.
        - **LabelSeletor**: (*Opcional*) filtra os recursos com escopo de cluster com base em seus rótulos.
      - **metadata.annotations.protect.trident.netapp.io/skip-vm-freeze**: (*Opcional*) Esta anotação só se aplica a aplicações definidas a partir de máquinas virtuais, como em ambientes KubeVirt, onde o congelamento do sistema de arquivos ocorre antes dos snapshots. Especifique se este aplicativo pode gravar no sistema de arquivos durante um snapshot. Se definido como verdadeiro, o aplicativo ignora a configuração global e pode gravar no sistema de arquivos durante um snapshot. Se definido como falso, o aplicativo ignora a configuração global e o sistema de arquivos é congelado durante a captura de um instantâneo. Se especificada, mas a aplicação não tiver máquinas virtuais na definição da aplicação, a anotação será ignorada. Caso não seja especificado, o aplicativo segue o"configuração de congelamento global do Trident Protect" .

Se você precisar aplicar essa anotação depois que um aplicativo já tiver sido criado, você pode usar o seguinte comando:

```
kubectl annotate application -n <application CR namespace> <application CR name> protect.trident.netapp.io/skip-vm-freeze="true"
```

+
Exemplo YAML:

+

```
apiVersion: protect.trident.netapp.io/v1
kind: Application
metadata:
  annotations:
    protect.trident.netapp.io/skip-vm-freeze: "false"
  name: my-app-name
  namespace: my-app-namespace
spec:
  includedNamespaces:
    - namespace: namespace-1
      labelSelector:
        matchLabels:
          app: example-app
    - namespace: namespace-2
      labelSelector:
        matchLabels:
          app: another-example-app
  includedClusterScopedResources:
    - groupVersionKind:
        group: rbac.authorization.k8s.io
        kind: ClusterRole
        version: v1
      labelSelector:
        matchLabels:
          mylabel: test
```

1. (*Opcional*) Adicione filtragem que inclua ou exclua recursos marcados com rótulos específicos:

    - **ResourceFilter.resourceSelectionCriteria**: (Necessário para filtragem) Use `Include` ou `Exclude` inclua ou exclua um recurso definido em resourceMatchers. Adicione os seguintes parâmetros resourceMatchers para definir os recursos a serem incluídos ou excluídos:

        - **ResourceFilter.resourceMatchers**: Uma matriz de resourceMatcher objetos. Se você definir vários elementos nesse array, eles corresponderão como uma OPERAÇÃO OU, e os campos dentro de cada elemento (grupo, tipo, versão) corresponderão como uma OPERAÇÃO E.
            - **ResourceMatchers[].group**: (*Optional*) Grupo do recurso a ser filtrado.
            - **ResourceMatchers[].kind**: (*Opcional*) tipo do recurso a ser filtrado.
            - **ResourceMatchers[].version**: (*Optional*) versão do recurso a ser filtrado.
            - **ResourceMatchers[].names**: (*Optional*) nomes no campo Kubernetes metadata.name do recurso a ser filtrado.

- **ResourceMatchers[].namespaces**: (*Optional*) namespaces no campo Kubernetes metadata.name do recurso a ser filtrado.
- **ResourceMatchers[].labelSelectors**: (*Optional*) string de seleção de etiquetas no campo Kubernetes metadata.name do recurso, conforme definido no "Documentação do Kubernetes". Por exemplo `"trident.netapp.io/os=linux":`.

> ⓘ Quando ambos `resourceFilter` e `labelSelector` são usados, `resourceFilter` corre primeiro e depois `labelSelector` é aplicado aos recursos resultantes.

Por exemplo:

```
spec:
  resourceFilter:
    resourceSelectionCriteria: "Include"
    resourceMatchers:
      - group: my-resource-group-1
        kind: my-resource-kind-1
        version: my-resource-version-1
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
      - group: my-resource-group-2
        kind: my-resource-kind-2
        version: my-resource-version-2
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
```

2. Depois de criar a aplicação CR para corresponder ao seu ambiente, aplique o CR. Por exemplo:

```
kubectl apply -f maria-app.yaml
```

Passos

1. Crie e aplique a definição do aplicativo usando um dos exemplos a seguir, substituindo valores entre parênteses por informações do ambiente. Você pode incluir namespaces e recursos na definição do aplicativo usando listas separadas por vírgulas com os argumentos mostrados nos exemplos.

   Você pode, opcionalmente, usar uma anotação ao criar um aplicativo para especificar se o aplicativo pode gravar no sistema de arquivos durante um snapshot. Isso só se aplica a aplicativos definidos a partir de máquinas virtuais, como em ambientes KubeVirt, onde o congelamento do sistema de arquivos ocorre antes dos snapshots. Se você definir a anotação para `true` , o aplicativo ignora a configuração global e pode gravar no sistema de arquivos durante um snapshot. Se você definir para `false` , o aplicativo ignora a configuração global e o sistema de arquivos fica congelado durante a captura de um snapshot. Se você usar a anotação, mas o aplicativo não tiver máquinas virtuais na definição do aplicativo, a anotação será ignorada. Se você não usar a anotação, o aplicativo seguirá

o padrão."configuração de congelamento global do Trident Protect" .

Para especificar a anotação quando você usa a CLI para criar um aplicativo, você pode usar o `--annotation` sinalizador.

- Crie o aplicativo e use 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 configuração do aplicativo local 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"]}]'
```

Proteja aplicativos usando o Trident Protect.

Você pode proteger todos os aplicativos gerenciados pelo Trident Protect criando snapshots e backups usando uma política de proteção automatizada ou de forma pontual.

Você pode configurar o Trident Protect para congelar e descongelar sistemas de arquivos durante operações de proteção de dados. "Saiba mais sobre como configurar o congelamento do sistema de arquivos com o Trident Protect.".

Crie 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 criado, configure os seguintes atributos:
   - **metadata.name**: (*required*) o nome deste recurso personalizado; escolha um nome único e sensível para o seu ambiente.
   - **Spec.applicationRef**: O nome do Kubernetes da aplicação para snapshot.
   - **Spec.appVaultRef**: (*required*) o nome do AppVault onde o conteúdo instantâneo (metadados) deve ser armazenado.
   - **Spec.reclaimPolicy**: (*Optional*) define o que acontece com o AppArchive de um snapshot quando o snapshot CR é excluído. Isso significa que, mesmo quando definido como `Retain`, o instantâneo será excluído. Opções válidas:
     - `Retain` (predefinição)
     - `Delete`

```
apiVersion: protect.trident.netapp.io/v1
kind: Snapshot
metadata:
  namespace: my-app-namespace
  name: my-cr-name
spec:
  applicationRef: my-application
  appVaultRef: appvault-name
  reclaimPolicy: Delete
```

3. Depois de preencher o `trident-protect-snapshot-cr.yaml` ficheiro com os valores corretos, aplique o CR:

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

Crie um instantâneo usando a CLI

Passos

1. Crie o snapshot, substituindo valores entre parênteses por informações do seu ambiente. Por exemplo:

```
tridentctl-protect create snapshot <my_snapshot_name> --appvault
<my_appvault_name> --app <name_of_app_to_snapshot> -n
<application_namespace>
```

Crie um backup sob demanda

Você pode fazer backup de um aplicativo a qualquer momento.

> Os recursos com escopo de cluster são incluídos em um backup, snapshot ou clone se forem explicitamente referenciados na definição do aplicativo ou se tiverem referências a qualquer um dos namespaces do aplicativo.

Antes de começar

Certifique-se de que a expiração do token de sessão da AWS seja suficiente para quaisquer operações de backup S3 de longa execução. Se o token expirar durante a operação de backup, a operação pode falhar.

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

Crie uma cópia de segurança utilizando um CR

Passos

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

     Exemplo YAML:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: Backup
metadata:
  namespace: my-app-namespace
  name: my-cr-name
spec:
  applicationRef: my-application
  appVaultRef: appvault-name
  dataMover: Kopia
```

3. Depois de preencher o `trident-protect-backup-cr.yaml` ficheiro com os valores corretos, aplique o CR:

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

Crie um backup usando a CLI

Passos

1. Crie o backup, substituindo valores entre parênteses por informações do seu ambiente. Por exemplo:

```
tridentctl-protect create backup <my_backup_name> --appvault <my-
vault-name> --app <name_of_app_to_back_up> --data-mover
<Kopia_or_Restic> -n <application_namespace>
```

Opcionalmente, você pode usar o `--full-backup` sinalizador para especificar se um backup deve ser não incremental. Por padrão, todos os backups são incrementais. Quando esse sinalizador é usado, o backup se torna não incremental. É prática recomendada executar um backup completo periodicamente e, em seguida, executar backups incrementais entre backups completos para minimizar o risco associado às restaurações.

Anotações de backup suportadas

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

Anotação	Tipo	Descrição	Valor padrão
protect.trident.netapp.io/backup completo	cadeia de carateres	Especifica se um backup deve ser não incremental. Definir para `true` Para criar um backup não incremental. A melhor prática é realizar um backup completo periodicamente e, em seguida, realizar backups incrementais entre os backups completos para minimizar o risco associado às restaurações.	"falso"
protect.trident.netapp.io/snapshot-completion-timeout	cadeia de carateres	Tempo máximo permitido para a conclusão da operação de captura instantânea.	"60m"
protect.trident.netapp.io/volume-snapshots-ready-to-use-timeout	cadeia de carateres	Tempo máximo permitido para que os snapshots de volume atinjam o estado pronto para uso.	"30m"
protect.trident.netapp.io/volume-snapshots-created-timeout	cadeia de carateres	Tempo máximo permitido para a criação de snapshots de volume.	"5m"
protect.trident.netapp.io/pvc-bind-timeout-sec	cadeia de carateres	Tempo máximo (em segundos) de espera para que quaisquer PersistentVolumeClaims (PVCs) recém-criados cheguem ao `Bound` fase anterior à falha das operações.	"1200" (20 minutos)

Criar um cronograma de proteção de dados

Uma política de proteção protege um aplicativo criando instantâneos, backups ou ambos em um cronograma definido. Você pode optar por criar snapshots e backups por hora, dia, semana e mês, e pode especificar o número de cópias a serem mantidas. Você pode agendar um backup completo não incremental usando a anotação full-backup-rule. Por padrão, todos os backups são incrementais. Executar um backup completo periodicamente, juntamente com backups incrementais entre eles, ajuda a reduzir o risco associado às restaurações.

- Você pode criar agendamentos apenas para instantâneos definindo `backupRetention` para zero e `snapshotRetention` para um valor maior que zero. Contexto `snapshotRetention` para zero significa que todos os backups agendados ainda criarão instantâneos, mas eles são temporários e serão excluídos imediatamente após a conclusão do backup.
- Os recursos com escopo de cluster são incluídos em um backup, snapshot ou clone se forem explicitamente referenciados na definição do aplicativo ou se tiverem referências a qualquer um dos namespaces do aplicativo.

Crie uma agenda usando um CR

Passos

1. Crie o arquivo de recurso personalizado (CR) e nomeie-o `trident-protect-schedule-cr.yaml`.
2. No arquivo criado, configure os seguintes atributos:
   - **metadata.name**: (*required*) o nome deste recurso personalizado; escolha um nome único e sensível para o seu ambiente.
   - **Spec.dataMover**: (*Optional*) Uma cadeia de carateres indicando qual ferramenta de backup usar para a operação de backup. Valores possíveis (sensíveis a maiúsculas e minúsculas):
     - `Restic`
     - `Kopia` (predefinição)
   - **Spec.applicationRef**: O nome do Kubernetes do aplicativo para fazer backup.
   - **Spec.appVaultRef**: (*required*) o nome do AppVault onde o conteúdo de backup deve ser armazenado.
   - **spec.backupRetention**: (*Obrigatório*) O número de backups a serem retidos. Zero indica que nenhum backup deve ser criado (somente instantâneos).
   - **spec.backupReclaimPolicy**: (Opcional) Determina o que acontece com um backup se o CR de backup for excluído durante seu período de retenção. Após o período de retenção, os backups são sempre apagados. Valores possíveis (diferencia maiúsculas de minúsculas):
     - `Retain` (predefinição)
     - `Delete`
   - **spec.snapshotRetention**: (*Obrigatório*) O número de snapshots a serem retidos. Zero indica que nenhum instantâneo deve ser criado.
   - **spec.snapshotReclaimPolicy**: (Opcional) Determina o que acontece com um snapshot se o CR do snapshot for excluído durante seu período de retenção. Após o período de retenção, os instantâneos são sempre apagados. Valores possíveis (diferencia maiúsculas de minúsculas):
     - `Retain`
     - `Delete`(padrão)
   - **spec.granularity**: a frequência em que o horário deve ser executado. Valores possíveis, juntamente com campos associados obrigatórios:
     - `Hourly`(requer que você especifique `spec.minute`)
     - `Daily`(requer que você especifique `spec.minute` e `spec.hour`)
     - `Weekly`(requer que você especifique `spec.minute, spec.hour` , e `spec.dayOfWeek`)
     - `Monthly`(requer que você especifique `spec.minute, spec.hour` , e `spec.dayOfMonth`)
     - `Custom`
   - **spec.dayOfMonth**: (*Opcional*) O dia do mês (1 - 31) em que o agendamento deve ser executado. Este campo é obrigatório se a granularidade estiver definida como `Monthly` . O valor deve ser fornecido como uma string.
   - **spec.dayOfWeek**: (*Opcional*) O dia da semana (0 - 7) em que a programação deve ser

executada. Valores de 0 ou 7 indicam domingo. Este campo é obrigatório se a granularidade estiver definida como `Weekly` . O valor deve ser fornecido como uma string.

- **spec.hour**: (*Opcional*) A hora do dia (0 - 23) em que a programação deve ser executada. Este campo é obrigatório se a granularidade estiver definida como `Daily` , `Weekly` , ou `Monthly` . O valor deve ser fornecido como uma string.
- **spec.minute**: (*Opcional*) O minuto da hora (0 - 59) em que a programação deve ser executada. Este campo é obrigatório se a granularidade estiver definida como `Hourly` , `Daily` , `Weekly` , ou `Monthly` . O valor deve ser fornecido como uma string.

Exemplo de YAML para agendamento de backup e snapshot:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: Schedule
metadata:
  namespace: my-app-namespace
  name: my-cr-name
spec:
  dataMover: Kopia
  applicationRef: my-application
  appVaultRef: appvault-name
  backupRetention: "15"
  snapshotRetention: "15"
  granularity: Daily
  hour: "0"
  minute: "0"
```

Exemplo de YAML para programação somente de snapshot:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: Schedule
metadata:
  namespace: my-app-namespace
  name: my-snapshot-schedule
spec:
  applicationRef: my-application
  appVaultRef: appvault-name
  backupRetention: "0"
  snapshotRetention: "15"
  granularity: Daily
  hour: "2"
  minute: "0"
```

3. Depois de preencher o `trident-protect-schedule-cr.yaml` ficheiro com os valores corretos,

aplique o CR:

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

Crie uma agenda usando a CLI

Passos

1. Crie o cronograma de proteção, substituindo valores entre parênteses por informações do seu ambiente. Por exemplo:

> Você pode usar `tridentctl-protect create schedule --help` para exibir informações detalhadas de ajuda para este comando.

```
tridentctl-protect create schedule <my_schedule_name> \
  --appvault <my_appvault_name> \
  --app <name_of_app_to_snapshot> \
  --backup-retention <how_many_backups_to_retain> \
  --backup-reclaim-policy <Retain|Delete (default Retain)> \
  --data-mover <Kopia_or_Restic> \
  --day-of-month <day_of_month_to_run_schedule> \
  --day-of-week <day_of_week_to_run_schedule> \
  --granularity <frequency_to_run> \
  --hour <hour_of_day_to_run> \
  --minute <minute_of_hour_to_run> \
  --recurrence-rule <recurrence> \
  --snapshot-retention <how_many_snapshots_to_retain> \
  --snapshot-reclaim-policy <Retain|Delete (default Delete)> \
  --full-backup-rule <string> \
  --run-immediately <true|false> \
  -n <application_namespace>
```

As seguintes opções oferecem controle adicional sobre sua programação:

- **Agendamento completo de backups**: Use o `--full-backup-rule` Sinalizador para agendar backups completos não incrementais. Esta flag só funciona com `--granularity Daily`. Valores possíveis:
  - `Always`Faça um backup completo todos os dias.
  - Dias da semana específicos: Especifique um ou mais dias separados por vírgulas (por exemplo, `"Monday,Thursday"`). Valores válidos: segunda-feira, terça-feira, quarta-feira, quinta-feira, sexta-feira, sábado, domingo.

> O `--full-backup-rule` A opção de sinalizar não funciona com granularidade horária, semanal ou mensal.

- **Agendamentos somente para instantâneos**: Definir `--backup-retention 0` e especifique um valor maior que zero para `--snapshot-retention`.

Anotações de agendamento suportadas

A tabela a seguir descreve as anotações que você pode usar ao criar uma solicitação de alteração (CR) de agendamento:

Anotação	Tipo	Descrição	Valor padrão
protect.trident.netapp.io/regra-de-backup-completo	cadeia de carateres	Especifica a regra para agendamento de backups completos. Você pode configurá-lo para `Always` Para backup completo constante ou personalize de acordo com suas necessidades. Por exemplo, se você escolher a granularidade diária, poderá especificar os dias da semana em que o backup completo deve ocorrer (por exemplo, `"Monday,Thursday"`). Os valores válidos para dias úteis são: segunda-feira, terça-feira, quarta-feira, quinta-feira, sexta-feira, sábado e domingo. Observe que esta anotação só pode ser usada com cronogramas que possuem `granularity` definido para `Daily`.	Não configurado (todos os backups são incrementais)
protect.trident.netapp.io/snapshot-completion-timeout	cadeia de carateres	Tempo máximo permitido para a conclusão da operação de captura instantânea.	"60m"
protect.trident.netapp.io/volume-snapshots-ready-to-use-timeout	cadeia de carateres	Tempo máximo permitido para que os snapshots de volume atinjam o estado pronto para uso.	"30m"
protect.trident.netapp.io/volume-snapshots-created-timeout	cadeia de carateres	Tempo máximo permitido para a criação de snapshots de volume.	"5m"
protect.trident.netapp.io/pvc-bind-timeout-sec	cadeia de carateres	Tempo máximo (em segundos) de espera para que quaisquer PersistentVolumeClaims (PVCs) recém-criados cheguem ao `Bound` fase anterior à falha das operações.	"1200" (20 minutos)

Eliminar um instantâneo

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

Passos

1. Remover o instantâneo CR associado ao instantâneo:

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

Eliminar uma cópia de segurança

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

Certifique-se de que a política de recuperação esteja definida como `Delete` para remover todos os dados de backup do armazenamento de objetos. A configuração padrão da política é `Retain` para evitar perda acidental de dados. Se a política não for alterada para `Delete` , os dados de backup permanecerão no armazenamento de objetos e exigirão exclusão manual.

Passos

1. Remova o CR de backup associado ao backup:

```
kubectl delete backup <backup_name> -n my-app-namespace
```

Verifique o status de uma operação de backup

Você pode usar a linha de comando para verificar o status de uma operação de backup em andamento, concluída ou falhou.

Passos

1. Use o seguinte comando para recuperar o status da operação de backup, substituindo valores em brackes por informações do seu ambiente:

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

Habilite o backup e a restauração de operações do Azure-NetApp-Files (ANF)

Se você instalou o Trident Protect, pode habilitar a funcionalidade de backup e restauração com uso eficiente de espaço para back-ends de armazenamento que utilizam a classe de armazenamento azure-netapp-files e foram criados antes do Trident 24.06. Essa funcionalidade opera com volumes NFSv4 e não consome espaço adicional do pool de capacidade.

Antes de começar

Certifique-se de que:

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

Expanda para obter as etapas de configuração

1. No Trident, se o volume do ANF tiver sido criado antes da atualização para o Trident 24,10:

   a. Ative o diretório instantâneo para cada PV que é baseado em azure-NetApp-Files e associado ao aplicativo:

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

   b. Confirme se o diretório instantâneo foi ativado para cada PV associado:

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

   Resposta:

   ```
   snapshotDirectory: "true"
   ```

   +
   Quando o diretório de instantâneos não está habilitado, o Trident Protect escolhe a funcionalidade de backup regular, que consome temporariamente espaço no pool de capacidade durante o processo de backup. Nesse caso, certifique-se de que haja espaço suficiente disponível no pool de capacidade para criar um volume temporário do tamanho do volume que está sendo copiado.

Resultado

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

Restaurar aplicativos

Restaure aplicativos usando o Trident Protect.

Você pode usar o Trident Protect para restaurar seu aplicativo a partir de um snapshot ou backup. A restauração a partir de um snapshot existente será mais rápida ao restaurar o aplicativo no mesmo cluster.

- Quando você restaura um aplicativo, todos os ganchos de execução configurados para o aplicativo são restaurados com o aplicativo. Se um gancho de execução pós-restauração estiver presente, ele será executado automaticamente como parte da operação de restauração.
- A restauração de um backup para um namespace diferente ou para o namespace original é suportada para volumes qtree. No entanto, a restauração de um snapshot para um namespace diferente ou para o namespace original não é suportada para volumes qtree.
- Você pode usar configurações avançadas para personalizar as operações de restauração. Para saber mais, consulte "Utilize as configurações avançadas de restauração do Trident Protect.".

Restaure de um backup para um namespace diferente

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

- Restaurar um backup para um namespace diferente com recursos existentes não alterará nenhum recurso que compartilhe nomes com aqueles no backup. Para restaurar todos os recursos no backup, exclua e recrie o namespace de destino ou restaure o backup para um novo namespace.
- Ao usar uma solicitação de restauração (CR) para restaurar em um novo namespace, você deve criar manualmente o namespace de destino antes de aplicar a CR. O Trident Protect cria namespaces automaticamente apenas quando se utiliza a CLI (linha de comando).

Antes de começar

Certifique-se de que a expiração do token de sessão da AWS seja suficiente para quaisquer operações de restauração S3 de longa duração. Se o token expirar durante a operação de restauração, a operação pode falhar.

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

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

Use um CR

Passos

1. Crie o arquivo de recurso personalizado (CR) e nomeie-o `trident-protect-backup-restore-cr.yaml`.
2. No arquivo criado, configure os seguintes atributos:
   - **metadata.name**: (*required*) o nome deste recurso personalizado; escolha um nome único e sensível para o seu ambiente.
   - **Spec.appArchivePath**: O caminho dentro do AppVault onde o conteúdo do backup é armazenado. Você pode usar o seguinte comando para encontrar este caminho:

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

   - **Spec.appVaultRef**: (*required*) o nome do AppVault onde o conteúdo de backup é armazenado.
   - **spec.namespaceMapping**: o mapeamento do namespace de origem da operação de restauração para o namespace de destino. Substitua `my-source-namespace` e `my-destination-namespace` por informações do seu ambiente.

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

3. (*Opcional*) se você precisar selecionar apenas determinados recursos do aplicativo para restaurar, adicione filtragem que inclua ou exclua recursos marcados com rótulos específicos:

   > O Trident Protect seleciona alguns recursos automaticamente devido à sua relação com recursos que você seleciona. Por exemplo, se você selecionar um recurso de reivindicação de volume persistente e ele tiver um pod associado, o Trident Protect também restaurará o pod associado.

   - **ResourceFilter.resourceSelectionCriteria**: (Necessário para filtragem) Use `Include` ou `Exclude` inclua ou exclua um recurso definido em resourceMatchers. Adicione os seguintes parâmetros resourceMatchers para definir os recursos a serem incluídos ou excluídos:
     - **ResourceFilter.resourceMatchers**: Uma matriz de resourceMatcher objetos. Se você definir vários elementos nesse array, eles corresponderão como uma OPERAÇÃO OU, e os campos dentro de cada elemento (grupo, tipo, versão) corresponderão como uma OPERAÇÃO E.

- **ResourceMatchers[].group**: (*Optional*) Grupo do recurso a ser filtrado.
- **ResourceMatchers[].kind**: (*Opcional*) tipo do recurso a ser filtrado.
- **ResourceMatchers[].version**: (*Optional*) versão do recurso a ser filtrado.
- **ResourceMatchers[].names**: (*Optional*) nomes no campo Kubernetes metadata.name do recurso a ser filtrado.
- **ResourceMatchers[].namespaces**: (*Optional*) namespaces no campo Kubernetes metadata.name do recurso a ser filtrado.
- **ResourceMatchers[].labelSelectors**: (*Optional*) string de seleção de etiquetas no campo Kubernetes metadata.name do recurso, conforme definido no "Documentação do Kubernetes". Por exemplo `"trident.netapp.io/os=linux":`.

  Por exemplo:

```
spec:
  resourceFilter:
    resourceSelectionCriteria: "Include"
    resourceMatchers:
      - group: my-resource-group-1
        kind: my-resource-kind-1
        version: my-resource-version-1
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
      - group: my-resource-group-2
        kind: my-resource-kind-2
        version: my-resource-version-2
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
```

4. Depois de preencher o `trident-protect-backup-restore-cr.yaml` ficheiro com os valores corretos, aplique o CR:

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

Use a CLI

Passos

1. Restaure o backup para um namespace diferente, substituindo valores entre parênteses por informações do seu ambiente. O `namespace-mapping` argumento usa namespaces separados por dois pontos para mapear namespaces de origem para os namespaces de destino corretos no formato `source1:dest1,source2:dest2`. Por exemplo:

```
tridentctl-protect create backuprestore <my_restore_name> \
--backup <backup_namespace>/<backup_to_restore> \
--namespace-mapping <source_to_destination_namespace_mapping> \
-n <application_namespace>
```

Restaure de um backup para o namespace original

Você pode restaurar um backup para o namespace original a qualquer momento.

Antes de começar

Certifique-se de que a expiração do token de sessão da AWS seja suficiente para quaisquer operações de restauração S3 de longa duração. Se o token expirar durante a operação de restauração, a operação pode falhar.

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

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

Use um CR

Passos

1. Crie o arquivo de recurso personalizado (CR) e nomeie-o `trident-protect-backup-ipr-cr.yaml`.
2. No arquivo criado, configure os seguintes atributos:
   - **metadata.name**: (*required*) o nome deste recurso personalizado; escolha um nome único e sensível para o seu ambiente.
   - **Spec.appArchivePath**: O caminho dentro do AppVault onde o conteúdo do backup é armazenado. Você pode usar o seguinte comando para encontrar este caminho:

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

   - **Spec.appVaultRef**: (*required*) o nome do AppVault onde o conteúdo de backup é armazenado.

     Por exemplo:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: BackupInplaceRestore
metadata:
  name: my-cr-name
  namespace: my-app-namespace
spec:
  appArchivePath: my-backup-path
  appVaultRef: appvault-name
```

3. (*Opcional*) se você precisar selecionar apenas determinados recursos do aplicativo para restaurar, adicione filtragem que inclua ou exclua recursos marcados com rótulos específicos:

   > O Trident Protect seleciona alguns recursos automaticamente devido à sua relação com recursos que você seleciona. Por exemplo, se você selecionar um recurso de reivindicação de volume persistente e ele tiver um pod associado, o Trident Protect também restaurará o pod associado.

   - **ResourceFilter.resourceSelectionCriteria**: (Necessário para filtragem) Use `Include` ou `Exclude` inclua ou exclua um recurso definido em resourceMatchers. Adicione os seguintes parâmetros resourceMatchers para definir os recursos a serem incluídos ou excluídos:
     - **ResourceFilter.resourceMatchers**: Uma matriz de resourceMatcher objetos. Se você definir vários elementos nesse array, eles corresponderão como uma OPERAÇÃO OU, e os campos dentro de cada elemento (grupo, tipo, versão) corresponderão como uma OPERAÇÃO E.
       - **ResourceMatchers[].group**: (*Optional*) Grupo do recurso a ser filtrado.
       - **ResourceMatchers[].kind**: (*Opcional*) tipo do recurso a ser filtrado.

- **ResourceMatchers[].version**: (*Optional*) versão do recurso a ser filtrado.
- **ResourceMatchers[].names**: (*Optional*) nomes no campo Kubernetes metadata.name do recurso a ser filtrado.
- **ResourceMatchers[].namespaces**: (*Optional*) namespaces no campo Kubernetes metadata.name do recurso a ser filtrado.
- **ResourceMatchers[].labelSelectors**: (*Optional*) string de seleção de etiquetas no campo Kubernetes metadata.name do recurso, conforme definido no "Documentação do Kubernetes". Por exemplo `"trident.netapp.io/os=linux"`: .

  Por exemplo:

```
spec:
  resourceFilter:
    resourceSelectionCriteria: "Include"
    resourceMatchers:
      - group: my-resource-group-1
        kind: my-resource-kind-1
        version: my-resource-version-1
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
      - group: my-resource-group-2
        kind: my-resource-kind-2
        version: my-resource-version-2
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
```

4. Depois de preencher o `trident-protect-backup-ipr-cr.yaml` ficheiro com os valores corretos, aplique o CR:

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

Use a CLI

Passos

1. Restaure o backup para o namespace original, substituindo valores entre parênteses por informações do seu ambiente. O `backup` argumento usa um namespace e um nome de backup no formato `<namespace>/<name>`. Por exemplo:

```
tridentctl-protect create backupinplacerestore <my_restore_name> \
--backup <namespace/backup_to_restore> \
-n <application_namespace>
```

Restaure de um backup para um cluster diferente

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

- Ao restaurar backups usando o Kopia como o movimentador de dados, você pode, opcionalmente, especificar anotações no CR ou usar a CLI para controlar o comportamento do armazenamento temporário usado pelo Kopia. Consulte o "Documentação da Kopia" Para obter mais informações sobre as opções que você pode configurar. Use o `tridentctl-protect create --help` Para obter mais informações sobre como especificar anotações com a CLI do Trident Protect, consulte o comando.
- Ao usar uma solicitação de restauração (CR) para restaurar em um novo namespace, você deve criar manualmente o namespace de destino antes de aplicar a CR. O Trident Protect cria namespaces automaticamente apenas quando se utiliza a CLI (linha de comando).

Antes de começar

Certifique-se de que os seguintes pré-requisitos são cumpridos:

- O cluster de destino tem o Trident Protect instalado.
- O cluster de destino tem acesso ao caminho do bucket do mesmo AppVault que o cluster de origem, onde o backup é armazenado.
- Ao executar o AppVault CR, certifique-se de que seu ambiente local possa se conectar ao bucket de armazenamento de objetos definido nele. `tridentctl-protect get appvaultcontent` comando. Caso as restrições de rede impeçam o acesso, execute a CLI do Trident Protect a partir de um pod no cluster de destino.
- Certifique-se de que a expiração do token de sessão da AWS seja suficiente para quaisquer operações de restauração de longa duração. Se o token expirar durante a operação de restauração, a operação pode falhar.
  - Consulte a "Documentação da API da AWS" para obter mais informações sobre como verificar a expiração do token de sessão atual.
  - Consulte o "Documentação do AWS" para obter mais informações sobre credenciais com recursos da AWS.

Passos

1. Verifique a disponibilidade do AppVault CR no cluster de destino usando o plugin Trident Protect CLI:

```
tridentctl-protect get appvault --context <destination_cluster_name>
```

Verifique se o namespace destinado à restauração do aplicativo existe no cluster de destino.

2. Veja o conteúdo de backup do AppVault disponível no cluster de destino:

```
tridentctl-protect get appvaultcontent <appvault_name> \
--show-resources backup \
--show-paths \
--context <destination_cluster_name>
```

Executar esse comando exibe os backups disponíveis no AppVault, incluindo os clusters de origem, nomes de aplicativos correspondentes, carimbos de data/hora e caminhos de arquivamento.

Exemplo de saída:

```
+-------------+-----------+--------+-----------------
+--------------------------+-------------+
|   CLUSTER   |    APP    |  TYPE  |      NAME       |        TIMESTAMP
|    PATH     |
+-------------+-----------+--------+-----------------
+--------------------------+-------------+
| production1 | wordpress | backup | wordpress-bkup-1| 2024-10-30
08:37:40 (UTC)| backuppath1 |
| production1 | wordpress | backup | wordpress-bkup-2| 2024-10-30
08:37:40 (UTC)| backuppath2 |
+-------------+-----------+--------+-----------------
+--------------------------+-------------+
```

3. Restaure o aplicativo para o cluster de destino usando o nome do AppVault e o caminho do arquivo:

Use um CR

1. Crie o arquivo de recurso personalizado (CR) e nomeie-o `trident-protect-backup-restore-cr.yaml`.
2. No arquivo criado, configure os seguintes atributos:
   - **metadata.name**: (*required*) o nome deste recurso personalizado; escolha um nome único e sensível para o seu ambiente.
   - **Spec.appVaultRef**: (*required*) o nome do AppVault onde o conteúdo de backup é armazenado.
   - **Spec.appArchivePath**: O caminho dentro do AppVault onde o conteúdo do backup é armazenado. Você pode usar o seguinte comando para encontrar este caminho:

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

     Se o BackupRestore CR não estiver disponível, você poderá usar o comando mencionado na etapa 2 para visualizar o conteúdo do backup.

   - **spec.namespaceMapping**: o mapeamento do namespace de origem da operação de restauração para o namespace de destino. Substitua `my-source-namespace` e `my-destination-namespace` por informações do seu ambiente.

     Por exemplo:

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

3. Depois de preencher o `trident-protect-backup-restore-cr.yaml` ficheiro com os valores corretos, aplique o CR:

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

Use a CLI

1. Use o comando a seguir para restaurar o aplicativo, substituindo valores entre parênteses por informações do ambiente. O argumento namespace-mapping usa namespaces separados por dois pontos para mapear namespaces de origem para os namespaces de destino corretos no formato

source1:dest1,source2:dest2. Por exemplo:

```
tridentctl-protect create backuprestore <restore_name> \
--namespace-mapping <source_to_destination_namespace_mapping> \
--appvault <appvault_name> \
--path <backup_path> \
--context <destination_cluster_name> \
-n <application_namespace>
```

Restauração de um snapshot para um namespace diferente

Você pode restaurar dados de um snapshot usando um arquivo de recurso personalizado (CR) para um namespace diferente ou para o namespace de origem original. Ao restaurar um snapshot para um namespace diferente usando um CR de SnapshotRestore, o Trident Protect restaura o aplicativo em um novo namespace e cria um CR de aplicativo para o aplicativo restaurado. Para proteger a aplicação restaurada, crie backups ou snapshots sob demanda, ou estabeleça um cronograma de proteção.

- O SnapshotRestore oferece suporte ao `spec.storageClassMapping` atributo, mas somente quando as classes de armazenamento de origem e destino usam o mesmo backend de armazenamento. Se você tentar restaurar para um `StorageClass` que usa um backend de armazenamento diferente, a operação de restauração falhará.
- Ao usar uma solicitação de restauração (CR) para restaurar em um novo namespace, você deve criar manualmente o namespace de destino antes de aplicar a CR. O Trident Protect cria namespaces automaticamente apenas quando se utiliza a CLI (linha de comando).

Antes de começar

Certifique-se de que a expiração do token de sessão da AWS seja suficiente para quaisquer operações de restauração S3 de longa duração. Se o token expirar durante a operação de restauração, a operação pode falhar.

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

Use um CR

Passos

1. Crie o arquivo de recurso personalizado (CR) e nomeie-o `trident-protect-snapshot-restore-cr.yaml`.
2. No arquivo criado, configure os seguintes atributos:
   - **metadata.name**: (*required*) o nome deste recurso personalizado; escolha um nome único e sensível para o seu ambiente.
   - **Spec.appVaultRef**: (*required*) o nome do AppVault onde o conteúdo do instantâneo é armazenado.
   - **Spec.appArchivePath**: O caminho dentro do AppVault onde o conteúdo do snapshot é armazenado. Você pode usar o seguinte comando para encontrar este caminho:

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

   - **spec.namespaceMapping**: o mapeamento do namespace de origem da operação de restauração para o namespace de destino. Substitua `my-source-namespace` e `my-destination-namespace` por informações do seu ambiente.

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

3. (*Opcional*) se você precisar selecionar apenas determinados recursos do aplicativo para restaurar, adicione filtragem que inclua ou exclua recursos marcados com rótulos específicos:

   > O Trident Protect seleciona alguns recursos automaticamente devido à sua relação com recursos que você seleciona. Por exemplo, se você selecionar um recurso de reivindicação de volume persistente e ele tiver um pod associado, o Trident Protect também restaurará o pod associado.

   - **ResourceFilter.resourceSelectionCriteria**: (Necessário para filtragem) Use `Include` ou `Exclude` inclua ou exclua um recurso definido em resourceMatchers. Adicione os seguintes parâmetros resourceMatchers para definir os recursos a serem incluídos ou excluídos:
     - **ResourceFilter.resourceMatchers**: Uma matriz de resourceMatcher objetos. Se você definir vários elementos nesse array, eles corresponderão como uma OPERAÇÃO OU, e os campos

dentro de cada elemento (grupo, tipo, versão) corresponderão como uma OPERAÇÃO E.

- **ResourceMatchers[].group**: (*Optional*) Grupo do recurso a ser filtrado.
- **ResourceMatchers[].kind**: (*Opcional*) tipo do recurso a ser filtrado.
- **ResourceMatchers[].version**: (*Optional*) versão do recurso a ser filtrado.
- **ResourceMatchers[].names**: (*Optional*) nomes no campo Kubernetes metadata.name do recurso a ser filtrado.
- **ResourceMatchers[].namespaces**: (*Optional*) namespaces no campo Kubernetes metadata.name do recurso a ser filtrado.
- **ResourceMatchers[].labelSelectors**: (*Optional*) string de seleção de etiquetas no campo Kubernetes metadata.name do recurso, conforme definido no "Documentação do Kubernetes". Por exemplo `"trident.netapp.io/os=linux"`:.

  Por exemplo:

```
spec:
  resourceFilter:
    resourceSelectionCriteria: "Include"
    resourceMatchers:
      - group: my-resource-group-1
        kind: my-resource-kind-1
        version: my-resource-version-1
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
      - group: my-resource-group-2
        kind: my-resource-kind-2
        version: my-resource-version-2
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
```

4. Depois de preencher o `trident-protect-snapshot-restore-cr.yaml` ficheiro com os valores corretos, aplique o CR:

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

Use a CLI

Passos

1. Restaure o snapshot para um namespace diferente, substituindo valores entre parênteses por informações do seu ambiente.

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

namespaces de origem para os namespaces de destino corretos no formato `source1:dest1,source2:dest2`.

Por exemplo:

```
tridentctl-protect create snapshotrestore <my_restore_name> \
--snapshot <namespace/snapshot_to_restore> \
--namespace-mapping <source_to_destination_namespace_mapping> \
-n <application_namespace>
```

Restauração de um snapshot para o namespace original

Você pode restaurar um snapshot para o namespace original a qualquer momento.

Antes de começar

Certifique-se de que a expiração do token de sessão da AWS seja suficiente para quaisquer operações de restauração S3 de longa duração. Se o token expirar durante a operação de restauração, a operação pode falhar.

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

Use um CR

Passos

1. Crie o arquivo de recurso personalizado (CR) e nomeie-o `trident-protect-snapshot-ipr-cr.yaml`.
2. No arquivo criado, configure os seguintes atributos:
   - **metadata.name**: (*required*) o nome deste recurso personalizado; escolha um nome único e sensível para o seu ambiente.
   - **Spec.appVaultRef**: (*required*) o nome do AppVault onde o conteúdo do instantâneo é armazenado.
   - **Spec.appArchivePath**: O caminho dentro do AppVault onde o conteúdo do snapshot é armazenado. Você pode usar o seguinte comando para encontrar este caminho:

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

```
---
apiVersion: protect.trident.netapp.io/v1
kind: SnapshotInplaceRestore
metadata:
  name: my-cr-name
  namespace: my-app-namespace
spec:
  appVaultRef: appvault-name
  appArchivePath: my-snapshot-path
```

3. (*Opcional*) se você precisar selecionar apenas determinados recursos do aplicativo para restaurar, adicione filtragem que inclua ou exclua recursos marcados com rótulos específicos:

> ⓘ O Trident Protect seleciona alguns recursos automaticamente devido à sua relação com recursos que você seleciona. Por exemplo, se você selecionar um recurso de reivindicação de volume persistente e ele tiver um pod associado, o Trident Protect também restaurará o pod associado.

   - **ResourceFilter.resourceSelectionCriteria**: (Necessário para filtragem) Use `Include` ou `Exclude` inclua ou exclua um recurso definido em resourceMatchers. Adicione os seguintes parâmetros resourceMatchers para definir os recursos a serem incluídos ou excluídos:
     - **ResourceFilter.resourceMatchers**: Uma matriz de resourceMatcher objetos. Se você definir vários elementos nesse array, eles corresponderão como uma OPERAÇÃO OU, e os campos dentro de cada elemento (grupo, tipo, versão) corresponderão como uma OPERAÇÃO E.
       - **ResourceMatchers[].group**: (*Optional*) Grupo do recurso a ser filtrado.
       - **ResourceMatchers[].kind**: (*Opcional*) tipo do recurso a ser filtrado.
       - **ResourceMatchers[].version**: (*Optional*) versão do recurso a ser filtrado.

- **ResourceMatchers[].names**: (*Optional*) nomes no campo Kubernetes metadata.name do recurso a ser filtrado.
- **ResourceMatchers[].namespaces**: (*Optional*) namespaces no campo Kubernetes metadata.name do recurso a ser filtrado.
- **ResourceMatchers[].labelSelectors**: (*Optional*) string de seleção de etiquetas no campo Kubernetes metadata.name do recurso, conforme definido no "Documentação do Kubernetes". Por exemplo `"trident.netapp.io/os=linux":.`

  Por exemplo:

```
spec:
  resourceFilter:
    resourceSelectionCriteria: "Include"
    resourceMatchers:
      - group: my-resource-group-1
        kind: my-resource-kind-1
        version: my-resource-version-1
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
      - group: my-resource-group-2
        kind: my-resource-kind-2
        version: my-resource-version-2
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
```

4. Depois de preencher o `trident-protect-snapshot-ipr-cr.yaml` ficheiro com os valores corretos, aplique o CR:

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

Use a CLI

Passos

1. Restaure o snapshot para o namespace original, substituindo valores entre parênteses por informações do seu ambiente. Por exemplo:

```
tridentctl-protect create snapshotinplacerestore <my_restore_name> \
--snapshot <namespace/snapshot_to_restore> \
-n <application_namespace>
```

Verifique o status de uma operação de restauração

Você pode usar a linha de comando para verificar o status de uma operação de restauração que está em andamento, concluiu ou falhou.

Passos

1. Use o seguinte comando para recuperar o status da operação de restauração, substituindo valores em brackes por informações do seu ambiente:

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

Utilize as configurações avançadas de restauração do Trident Protect.

Você pode personalizar operações de restauração usando configurações avançadas, como anotações, configurações de namespace e opções de armazenamento para atender aos seus requisitos específicos.

Anotações e rótulos de namespace durante operações de restauração e failover

Durante as operações de restauração e failover, rótulos e anotações no namespace de destino são feitos para corresponder aos rótulos e anotações no namespace de origem. Rótulos ou anotações do namespace de origem que não existem no namespace de destino são adicionados, e quaisquer rótulos ou anotações que já existem são sobrescritos para corresponder ao valor do namespace de origem. Rótulos ou anotações que existem apenas no namespace de destino permanecem inalterados.

> Se você usa o Red Hat OpenShift, é importante observar o papel crítico das anotações de namespace em ambientes OpenShift. As anotações de namespace garantem que os pods restaurados cumpram as permissões e configurações de segurança apropriadas definidas pelas restrições de contexto de segurança (SCCs) do OpenShift e possam acessar volumes sem problemas de permissão. Para mais informações, consulte o"Documentação de restrições de contexto de segurança OpenShift" .

Você pode impedir que anotações específicas no namespace de destino sejam sobrescritas definindo a variável de ambiente do Kubernetes `RESTORE_SKIP_NAMESPACE_ANNOTATIONS` antes de executar a operação de restauração ou failover. Por exemplo:

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect \
  --set-string
restoreSkipNamespaceAnnotations="{<annotation_key_to_skip_1>,<annotation_k
ey_to_skip_2>}" \
  --reuse-values
```

Ao executar uma operação de restauração ou failover, quaisquer anotações e rótulos de namespace especificados em `restoreSkipNamespaceAnnotations` e `restoreSkipNamespaceLabels` são excluídos da operação de restauração ou failover. Certifique-se de que essas configurações sejam definidas durante a instalação inicial do Helm. Para saber mais, consulte "Configure as definições adicionais do gráfico de navegação do Trident Protect.".

Se você instalou o aplicativo de origem usando o Helm com o `--create-namespace` bandeira, tratamento especial é dado ao `name` Legenda da etiqueta. Durante o processo de restauração ou failover, o Trident Protect copia esse rótulo para o namespace de destino, mas atualiza o valor para o valor do namespace de destino se o valor da origem corresponder ao namespace de origem. Se esse valor não corresponder ao namespace de origem, ele será copiado para o namespace de destino sem alterações.

Exemplo

O exemplo a seguir apresenta um namespace de origem e destino, cada um com anotações e rótulos diferentes. Você pode ver o estado do namespace de destino antes e depois da operação e como as anotações e rótulos são combinados ou substituídos no namespace de destino.

Antes da operação de restauração ou failover

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

Namespace	Anotações	Etiquetas
Namespace ns-1 (fonte)	• annotation.one/key: "updatedvalue" • annotation.two/key: "true"	• ambiente de produção • conformidade hipaa • nome: ns-1
Namespace ns-2 (destino)	• annotation.one/key: "true" (verdadeiro) • annotation.three/key: "false"	• banco de dados

Após a operação de restauração

A tabela a seguir ilustra o estado do namespace de destino de exemplo após a operação de restauração ou failover. Algumas chaves foram adicionadas, algumas foram sobrescritas e o `name` rótulo foi atualizado para corresponder ao namespace de destino:

Namespace	Anotações	Etiquetas
Namespace ns-2 (destino)	• annotation.one/key: "updatedvalue" • annotation.two/key: "true" • annotation.three/key: "false"	• nome: ns-2 • conformidade hipaa • ambiente de produção • banco de dados

Campos suportados

Esta seção descreve campos adicionais disponíveis para operações de restauração.

Mapeamento de classes de armazenamento

O `spec.storageClassMapping` atributo define um mapeamento de uma classe de armazenamento presente no aplicativo de origem para uma nova classe de armazenamento no cluster de destino. Você pode usar isso ao migrar aplicativos entre clusters com diferentes classes de armazenamento ou ao alterar o backend de armazenamento para operações de BackupRestore.

Exemplo:

```
storageClassMapping:
  - destination: "destinationStorageClass1"
    source: "sourceStorageClass1"
  - destination: "destinationStorageClass2"
    source: "sourceStorageClass2"
```

Anotações suportadas

Esta seção lista as anotações suportadas para configurar diversos comportamentos no sistema. Se uma anotação não for definida explicitamente pelo usuário, o sistema usará o valor padrão.

Anotação	Tipo	Descrição	Valor padrão
protect.trident.netapp.io/data-mover-timeout-sec	cadeia de carateres	O tempo máximo (em segundos) permitido para a operação do movimentador de dados ser interrompida.	"300"
protect.trident.netapp.io/kopia-content-cache-size-limit-mb	cadeia de carateres	O limite máximo de tamanho (em megabytes) para o cache de conteúdo do Kopia.	"1000"
protect.trident.netapp.io/pvc-bind-timeout-sec	cadeia de carateres	Tempo máximo (em segundos) de espera para que quaisquer PersistentVolumeClaims (PVCs) recém-criados cheguem ao `Bound` fase anterior à falha das operações. Aplica-se a todos os tipos de restauração de CR (BackupRestore, BackupInplaceRestore, SnapshotRestore, SnapshotInplaceRestore). Use um valor mais alto se o seu sistema de armazenamento ou cluster frequentemente exigir mais tempo.	"1200" (20 minutos)

Replique aplicações usando NetApp SnapMirror e Trident Protect.

Com o Trident Protect, você pode usar os recursos de replicação assíncrona da tecnologia NetApp SnapMirror para replicar dados e alterações de aplicativos de um backend de armazenamento para outro, no mesmo cluster ou entre clusters diferentes.

Anotações e rótulos de namespace durante operações de restauração e failover

Durante as operações de restauração e failover, rótulos e anotações no namespace de destino são feitos para corresponder aos rótulos e anotações no namespace de origem. Rótulos ou anotações do namespace de origem que não existem no namespace de destino são adicionados, e quaisquer rótulos ou anotações que já existem são sobrescritos para corresponder ao valor do namespace de origem. Rótulos ou anotações que existem apenas no namespace de destino permanecem inalterados.

> Se você usa o Red Hat OpenShift, é importante observar o papel crítico das anotações de namespace em ambientes OpenShift. As anotações de namespace garantem que os pods restaurados cumpram as permissões e configurações de segurança apropriadas definidas pelas restrições de contexto de segurança (SCCs) do OpenShift e possam acessar volumes sem problemas de permissão. Para mais informações, consulte o"Documentação de restrições de contexto de segurança OpenShift" .

Você pode impedir que anotações específicas no namespace de destino sejam sobrescritas definindo a variável de ambiente do Kubernetes `RESTORE_SKIP_NAMESPACE_ANNOTATIONS` antes de executar a operação de restauração ou failover. Por exemplo:

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect \
  --set-string
restoreSkipNamespaceAnnotations="{<annotation_key_to_skip_1>,<annotation_k
ey_to_skip_2>}" \
  --reuse-values
```

> Ao executar uma operação de restauração ou failover, quaisquer anotações e rótulos de namespace especificados em `restoreSkipNamespaceAnnotations` e `restoreSkipNamespaceLabels` são excluídos da operação de restauração ou failover. Certifique-se de que essas configurações sejam definidas durante a instalação inicial do Helm. Para saber mais, consulte "Configure as definições adicionais do gráfico de navegação do Trident Protect.".

Se você instalou o aplicativo de origem usando o Helm com o `--create-namespace` bandeira, tratamento especial é dado ao `name` Legenda da etiqueta. Durante o processo de restauração ou failover, o Trident Protect copia esse rótulo para o namespace de destino, mas atualiza o valor para o valor do namespace de destino se o valor da origem corresponder ao namespace de origem. Se esse valor não corresponder ao namespace de origem, ele será copiado para o namespace de destino sem alterações.

Exemplo

O exemplo a seguir apresenta um namespace de origem e destino, cada um com anotações e rótulos diferentes. Você pode ver o estado do namespace de destino antes e depois da operação e como as anotações e rótulos são combinados ou substituídos no namespace de destino.

Antes da operação de restauração ou failover

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

Namespace	Anotações	Etiquetas
Namespace ns-1 (fonte)	• annotation.one/key: "updatedvalue" • annotation.two/key: "true"	• ambiente de produção • conformidade hipaa • nome: ns-1
Namespace ns-2 (destino)	• annotation.one/key: "true" (verdadeiro) • annotation.three/key: "false"	• banco de dados

Após a operação de restauração

A tabela a seguir ilustra o estado do namespace de destino de exemplo após a operação de restauração ou failover. Algumas chaves foram adicionadas, algumas foram sobrescritas e o `name` rótulo foi atualizado para corresponder ao namespace de destino:

Namespace	Anotações	Etiquetas
Namespace ns-2 (destino)	• annotation.one/key: "updatedvalue" • annotation.two/key: "true" • annotation.three/key: "false"	• nome: ns-2 • conformidade hipaa • ambiente de produção • banco de dados

Você pode configurar o Trident Protect para congelar e descongelar sistemas de arquivos durante operações de proteção de dados. "Saiba mais sobre como configurar o congelamento do sistema de arquivos com o Trident Protect.".

Ganchos de execução durante operações de failover e reverso

Ao usar o relacionamento do AppMirror para proteger seu aplicativo, há comportamentos específicos relacionados aos ganchos de execução dos quais você deve estar ciente durante operações de failover e reverso.

- Durante o failover, os ganchos de execução são copiados automaticamente do cluster de origem para o cluster de destino. Não é necessário recriá-los manualmente. Após o failover, os ganchos de execução permanecem presentes no aplicativo e serão executados durante quaisquer ações relevantes.
- Durante a sincronização reversa ou ressincronização reversa, quaisquer ganchos de execução existentes no aplicativo são removidos. Quando o aplicativo de origem se torna o aplicativo de destino, esses ganchos de execução não são válidos e são excluídos para impedir sua execução.

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

Configure uma relação de replicação

A configuração de uma relação de replicação envolve o seguinte:

- Escolher a frequência com que o Trident Protect deve tirar um snapshot do aplicativo (que inclui os recursos do Kubernetes do aplicativo, bem como os snapshots de volume para cada um dos volumes do

aplicativo).

- Escolha do cronograma de replicação (inclui recursos do Kubernetes e dados de volume persistente)
- Definir o tempo para a captura instantânea

Passos

1. No cluster de origem, crie um AppVault para o aplicativo de origem. Dependendo do seu fornecedor de storage, modifique um exemplo no "Recursos personalizados do AppVault" para se adequar ao seu ambiente:

Crie um AppVault usando um CR

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

b. Configure os seguintes atributos:

- **metadata.name**: (*required*) o nome do recurso personalizado do AppVault. Anote o nome que você escolher, porque outros arquivos CR necessários para uma relação de replicação referem-se a esse valor.
- **spec.providerConfig**: (*required*) armazena a configuração necessária para acessar o AppVault usando o provedor especificado. Escolha um bucketName e quaisquer outros detalhes necessários para o seu provedor. Anote os valores que você escolher, porque outros arquivos CR necessários para uma relação de replicação se referem a esses valores. "Recursos personalizados do AppVault"Consulte para obter exemplos de AppVault CRS com outros provedores.
- **spec.providerCredentials**: (*required*) armazena referências a qualquer credencial necessária para acessar o AppVault usando o provedor especificado.
  - **spec.providerCredentials.valueFromSecret**: (*required*) indica que o valor da credencial deve vir de um segredo.
    - **Key**: (*required*) a chave válida do segredo para selecionar.
    - **Name**: (*required*) Nome do segredo que contém o valor deste campo. Deve estar no mesmo namespace.
  - **spec.providerCredentials.secretAccessKey**: (*required*) a chave de acesso usada para acessar o provedor. O **nome** deve corresponder a **spec.providerCredentials.valueFromSecret.name**.
- **spec.providerType**: (*required*) determina o que fornece o backup; por exemplo, NetApp ONTAP S3, S3 genérico, Google Cloud ou Microsoft Azure. Valores possíveis:
  - aws
  - azure
  - gcp
  - generic-s3
  - ONTAP-s3
  - StorageGRID-s3

c. Depois de preencher o `trident-protect-appvault-primary-source.yaml` ficheiro com os valores corretos, aplique o CR:

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

Crie um AppVault usando a CLI

a. Crie o AppVault, substituindo valores entre parênteses por informações do seu ambiente:

```
tridentctl-protect create vault Azure <vault-name> --account
<account-name> --bucket <bucket-name> --secret <secret-name> -n
trident-protect
```

2. No cluster de origem, crie a aplicação de origem CR:

Crie o aplicativo de origem usando um CR

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

b. Configure os seguintes atributos:

- **metadata.name**: (*required*) o nome do recurso personalizado do aplicativo. Anote o nome que você escolher, porque outros arquivos CR necessários para uma relação de replicação referem-se a esse valor.
- **spec.includedNamespaces**: (*required*) um array de namespaces e rótulos associados. Use nomes de namespace e, opcionalmente, restrinja o escopo dos namespaces com rótulos para especificar recursos que existem nos namespaces listados aqui. O namespace da aplicação deve fazer parte desse array.

  Exemplo YAML:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: Application
metadata:
  name: my-app-name
  namespace: my-app-namespace
spec:
  includedNamespaces:
    - namespace: my-app-namespace
      labelSelector: {}
```

c. Depois de preencher o `trident-protect-app-source.yaml` ficheiro com os valores corretos, aplique o CR:

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

Crie o aplicativo de origem usando a CLI

a. Crie o aplicativo de origem. Por exemplo:

```
tridentctl-protect create app <my-app-name> --namespaces
<namespaces-to-be-included> -n <my-app-namespace>
```

3. Opcionalmente, no cluster de origem, tire um snapshot do aplicativo de origem. Este instantâneo é usado como base para a aplicação no cluster de destino. Se você pular esta etapa, precisará aguardar a próxima captura de instantâneo agendada para obter um instantâneo recente. Para criar um instantâneo sob demanda, consulte "Crie um snapshot sob demanda".

4. No cluster de origem, crie a solicitação de configuração (CR) para o agendamento de replicação:

Além do cronograma fornecido abaixo, recomenda-se criar um cronograma de snapshots diários separado, com um período de retenção de 7 dias, para manter um snapshot comum entre clusters ONTAP pareados. Isso garante que os snapshots fiquem disponíveis por até 7 dias, mas o período de retenção pode ser personalizado de acordo com as necessidades do usuário.

Em caso de failover, o sistema pode usar esses snapshots por até 7 dias para operações reversas. Essa abordagem torna o processo de reversão mais rápido e eficiente, pois apenas as alterações feitas desde o último snapshot serão transferidas, e não todos os dados.

Se um cronograma existente para o aplicativo já atender aos requisitos de retenção desejados, nenhum cronograma adicional será necessário.

Crie o cronograma de replicação usando uma CR (Código de Referência).

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

i. Crie o arquivo de recurso personalizado (CR) e nomeie-o (por exemplo, `trident-protect-schedule.yaml`).

ii. Configure os seguintes atributos:

- **metadata.name**: (*required*) o nome do recurso personalizado de agendamento.
- **spec.appVaultRef**: (*Obrigatório*) Este valor deve corresponder ao campo metadata.name do AppVault para o aplicativo de origem.
- **spec.applicationRef**: (*Obrigatório*) Este valor deve corresponder ao campo metadata.name do CR da aplicação de origem.
- **Spec.backupRetention**: (*required*) este campo é obrigatório e o valor deve ser definido como 0.
- **Spec.enabled**: Deve ser definido como true.
- **spec.granularity**: tem de estar definido para `Custom`.
- **Spec.recurrenceRule**: Defina uma data de início no horário UTC e um intervalo de recorrência.
- **Spec.snapshotRetention**: Deve ser definido como 2.

Exemplo YAML:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: Schedule
metadata:
  name: appmirror-schedule
  namespace: my-app-namespace
spec:
  appVaultRef: my-appvault-name
  applicationRef: my-app-name
  backupRetention: "0"
  enabled: true
  granularity: Custom
  recurrenceRule: |-
    DTSTART:20220101T000200Z
    RRULE:FREQ=MINUTELY;INTERVAL=5
  snapshotRetention: "2"
```

i. Depois de preencher o `trident-protect-schedule.yaml` ficheiro com os valores corretos, aplique o CR:

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

Crie o agendamento de replicação usando a CLI

a. Crie o cronograma de replicação, substituindo os valores entre colchetes pelas informações do seu ambiente:

```
tridentctl-protect create schedule --name appmirror-schedule
--app <my_app_name> --appvault <my_app_vault> --granularity
Custom --recurrence-rule <rule> --snapshot-retention
<snapshot_retention_count> -n <my_app_namespace>
```

Exemplo:

```
tridentctl-protect create schedule --name appmirror-schedule
--app <my_app_name> --appvault <my_app_vault> --granularity
Custom --recurrence-rule  "DTSTART:20220101T000200Z
\nRRULE:FREQ=MINUTELY;INTERVAL=5" --snapshot-retention 2 -n
<my_app_namespace>
```

5. No cluster de destino, crie um aplicativo de origem AppVault CR idêntico ao AppVault CR aplicado no cluster de origem e nomeie-o (por exemplo, `trident-protect-appvault-primary-destination.yaml`).
6. Aplicar o CR:

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

7. Crie um AppVault CR de destino para o aplicativo de destino no cluster de destino. Dependendo do seu fornecedor de storage, modifique um exemplo no "Recursos personalizados do AppVault" para se adequar ao seu ambiente:
   a. Crie o arquivo de recurso personalizado (CR) e nomeie-o (por exemplo, `trident-protect-appvault-secondary-destination.yaml`).
   b. Configure os seguintes atributos:
      - **metadata.name**: (*required*) o nome do recurso personalizado do AppVault. Anote o nome que você escolher, porque outros arquivos CR necessários para uma relação de replicação referem-se a esse valor.
      - **spec.providerConfig**: (*required*) armazena a configuração necessária para acessar o AppVault usando o provedor especificado. Escolha um `bucketName` e quaisquer outros detalhes necessários para o seu provedor. Anote os valores que você escolher, porque outros arquivos CR necessários para uma relação de replicação se referem a esses valores. "Recursos

personalizados do AppVault"Consulte para obter exemplos de AppVault CRS com outros provedores.

- **spec.providerCredentials**: (*required*) armazena referências a qualquer credencial necessária para acessar o AppVault usando o provedor especificado.
  - **spec.providerCredentials.valueFromSecret**: (*required*) indica que o valor da credencial deve vir de um segredo.
    - **Key**: (*required*) a chave válida do segredo para selecionar.
    - **Name**: (*required*) Nome do segredo que contém o valor deste campo. Deve estar no mesmo namespace.
  - **spec.providerCredentials.secretAccessKey**: (*required*) a chave de acesso usada para acessar o provedor. O **nome** deve corresponder a **spec.providerCredentials.valueFromSecret.name**.
- **spec.providerType**: (*required*) determina o que fornece o backup; por exemplo, NetApp ONTAP S3, S3 genérico, Google Cloud ou Microsoft Azure. Valores possíveis:
  - aws
  - azure
  - gcp
  - generic-s3
  - ONTAP-s3
  - StorageGRID-s3

c. Depois de preencher o `trident-protect-appvault-secondary-destination.yaml` ficheiro com os valores corretos, aplique o CR:

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

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

> Ao usar uma solicitação de alteração (CR), crie manualmente o namespace de destino antes de aplicar a CR. O Trident Protect cria namespaces automaticamente apenas quando se utiliza a CLI (linha de comando).

Crie um AppMirrorRelationship usando um CR

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

b. Configure os seguintes atributos:

- **metadata.name:** (obrigatório) o nome do recurso personalizado AppMirrorRelationship.
- **spec.destinationAppVaultRef**: (*required*) esse valor deve corresponder ao nome do AppVault para o aplicativo de destino no cluster de destino.
- **spec.namespaceMapping**: (*required*) os namespaces de destino e origem devem corresponder ao namespace de aplicativo definido no respetivo CR de aplicação.
- **Spec.sourceAppVaultRef**: (*required*) este valor deve corresponder ao nome do AppVault para o aplicativo de origem.
- **Spec.sourceApplicationName**: (*required*) esse valor deve corresponder ao nome do aplicativo de origem definido no CR do aplicativo de origem.
- **spec.sourceApplicationUID**: (Obrigatório) Este valor deve corresponder ao UID do aplicativo de origem que você definiu no CR do aplicativo de origem.
- **spec.storageClassName**: (Opcional) Escolha o nome de uma classe de armazenamento válida no cluster. A classe de armazenamento deve ser vinculada a uma VM de armazenamento ONTAP que esteja emparelhada com o ambiente de origem. Caso a classe de armazenamento não seja especificada, a classe de armazenamento padrão do cluster será utilizada por padrão.
- **Spec.recurrenceRule**: Defina uma data de início no horário UTC e um intervalo de recorrência.

  Exemplo YAML:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: AppMirrorRelationship
metadata:
  name: amr-16061e80-1b05-4e80-9d26-d326dc1953d8
  namespace: my-app-namespace
spec:
  desiredState: Established
  destinationAppVaultRef: generic-s3-trident-protect-dst-bucket-
8fe0b902-f369-4317-93d1-ad7f2edc02b5
  namespaceMapping:
    - destination: my-app-namespace
      source: my-app-namespace
  recurrenceRule: |-
    DTSTART:20220101T000200Z
    RRULE:FREQ=MINUTELY;INTERVAL=5
  sourceAppVaultRef: generic-s3-trident-protect-src-bucket-
b643cc50-0429-4ad5-971f-ac4a83621922
  sourceApplicationName: my-app-name
  sourceApplicationUID: 7498d32c-328e-4ddd-9029-122540866aeb
  storageClassName: sc-vsim-2
```

c. Depois de preencher o `trident-protect-relationship.yaml` ficheiro com os valores corretos, aplique o CR:

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

Crie um AppMirrorRelationship usando a CLI

a. Crie e aplique o objeto AppMirrorRelationship, substituindo os valores entre colchetes pelas informações do seu ambiente:

```
tridentctl-protect create appmirrorrelationship
<name_of_appmirorrelationship> --destination-app-vault
<my_vault_name> --source-app-vault <my_vault_name> --recurrence
-rule <rule> --namespace-mapping <ns_mapping> --source-app-id
<source_app_UID> --source-app <my_source_app_name> --storage
-class <storage_class_name> -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. (*Optional*) no cluster de destino, verifique o estado e o estado da relação de replicação:

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

Failover para o cluster de destino

Com o Trident Protect, você pode realizar o failover de aplicativos replicados para um cluster de destino. Este procedimento interrompe a relação de replicação e coloca o aplicativo online no cluster de destino. O Trident Protect não interrompe o aplicativo no cluster de origem se ele estiver operacional.

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. Aplicar o CR:

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

4. (*Optional*) Crie todos os programas de proteção que você precisa no aplicativo com falha.
5. (*Optional*) Verifique o estado e o estado da relação de replicação:

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

Ressincronizar uma relação de replicação com falha

A operação ressincronizada restabelece a relação de replicação. Depois de executar uma operação ressincronizada, o aplicativo de origem original se torna o aplicativo em execução e quaisquer alterações feitas no aplicativo em execução no cluster de destino serão descartadas.

O processo pára o aplicativo no cluster de destino antes de restabelecer a replicação.

Todos os dados gravados na aplicação de destino durante o failover serão perdidos.

Passos

1. Opcional: No cluster de origem, crie um instantâneo do aplicativo de origem. Isso garante que as alterações mais recentes do cluster de origem sejam capturadas.
2. No cluster de destino, edite o arquivo CR AppMirrorRelationship (por exemplo, `trident-protect-relationship.yaml`) e altere o valor de spec.desiredState para `Established`.
3. Salve o arquivo CR.
4. Aplicar o CR:

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

5. Se você criou quaisquer programações de proteção no cluster de destino para proteger o aplicativo com falha, remova-os. Quaisquer programações restantes causam falhas de snapshot de volume.

Ressincronização reversa de uma relação de replicação com falha

Quando você faz a ressincronização reversa de uma relação de replicação com falha, o aplicativo de destino se torna o aplicativo de origem e a origem se torna o destino. As alterações feitas na aplicação de destino durante o failover são mantidas.

Passos

1. No cluster de destino original, exclua o AppMirrorRelationship CR. Isso faz com que o destino se torne a fonte. Se houver planos de proteção restantes no novo cluster de destino, remova-os.
2. Configure uma relação de replicação aplicando os arquivos CR usados originalmente para configurar a relação com os clusters opostos.
3. Certifique-se de que o novo destino (cluster de origem original) esteja configurado com o AppVault CRS.
4. Configure uma relação de replicação no cluster oposto, configurando valores para a direção inversa.

Sentido de replicação da aplicação inversa

Ao inverter a direção da replicação, o Trident Protect move a aplicação para o backend de armazenamento de destino, enquanto continua a replicar de volta para o backend de armazenamento de origem original. O Trident Protect interrompe o aplicativo de origem e replica os dados para o destino antes de realizar o failover para o aplicativo de destino.

Nesta situação, você está trocando a origem e o destino.

Passos

1. No cluster de origem, crie um instantâneo de encerramento:

Crie um instantâneo de encerramento utilizando um CR

a. Desative as programações de políticas de proteção para o aplicativo de origem.

b. Criar um ficheiro ShutdownSnapshot CR:

   i. Crie o arquivo de recurso personalizado (CR) e nomeie-o (por exemplo, `trident-protect-shutdownsnapshot.yaml`).

   ii. Configure os seguintes atributos:

   - **metadata.name**: (*required*) o nome do recurso personalizado.
   - **Spec.AppVaultRef**: (*required*) este valor deve corresponder ao campo metadata.name do AppVault para o aplicativo de origem.
   - **Spec.ApplicationRef**: (*required*) este valor deve corresponder ao campo metadata.name do arquivo CR da aplicação de origem.

     Exemplo YAML:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: ShutdownSnapshot
metadata:
  name: replication-shutdown-snapshot-afc4c564-e700-4b72-86c3-
c08a5dbe844e
  namespace: my-app-namespace
spec:
  appVaultRef: generic-s3-trident-protect-src-bucket-04b6b4ec-
46a3-420a-b351-45795e1b5e34
  applicationRef: my-app-name
```

c. Depois de preencher o `trident-protect-shutdownsnapshot.yaml` ficheiro com os valores corretos, aplique o CR:

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

Crie um instantâneo de encerramento usando a CLI

a. Crie o instantâneo de encerramento, substituindo valores entre parênteses por informações do seu ambiente. Por exemplo:

```
tridentctl-protect create shutdownsnapshot <my_shutdown_snapshot>
--appvault <my_vault> --app <app_to_snapshot> -n
<application_namespace>
```

2. No cluster de origem, após a conclusão do instantâneo de encerramento, obtenha o status do instantâneo de encerramento:

```
kubectl get shutdownsnapshot -n my-app-namespace
<shutdown_snapshot_name> -o yaml
```

3. No cluster de origem, encontre o valor de **shutdownsnapshot.status.appArchivePath** usando o seguinte comando, e Registre a última parte do caminho do arquivo (também chamado de basename; isso será tudo após a última barra):

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

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

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

5. Execute as etapas de ressincronização reversa no Ressincronização reversa de uma relação de replicação com falha.
6. Ative programações de proteção no novo cluster de origem.

Resultado

As seguintes ações ocorrem devido à replicação reversa:

- Um snapshot é obtido dos recursos do Kubernetes do aplicativo de origem original.
- Os pods do aplicativo de origem original são interrompidos graciosamente ao excluir os recursos do Kubernetes do aplicativo (deixando PVCs e PVS no lugar).
- Depois que os pods são desativados, snapshots dos volumes do aplicativo são feitos e replicados.
- As relações do SnapMirror são quebradas, tornando os volumes de destino prontos para leitura/gravação.
- Os recursos do Kubernetes do aplicativo são restaurados a partir do snapshot de pré-encerramento, usando os dados de volume replicados após o desligamento do aplicativo de origem original.
- A replicação é restabelecida na direção inversa.

Falha de aplicativos para o cluster de origem original

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

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

- Comece com um estado com falha em excesso.

- Reverta a ressincronização da relação de replicação.

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

- Inverta a direção da replicação.

Passos

1. Execute os Ressincronização reversa de uma relação de replicação com falha passos.
2. Execute os Sentido de replicação da aplicação inversa passos.

Excluir uma relação de replicação

Você pode excluir um relacionamento de replicação a qualquer momento. Quando você exclui a relação de replicação do aplicativo, isso resulta em dois aplicativos separados sem relação entre eles.

Passos

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

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

Migre aplicativos usando o Trident Protect.

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

> Quando você migra um aplicativo, todos os ganchos de execução configurados para o aplicativo são migrados com o aplicativo. Se um gancho de execução pós-restauração estiver presente, ele será executado automaticamente como parte da operação de restauração.

Operações de backup e restauração

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

Clonar para o mesmo cluster

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

Passos

1. Execute um dos seguintes procedimentos:
   a. "Criar um instantâneo".
   b. "Crie uma cópia de segurança".
2. No mesmo cluster, siga um destes procedimentos, dependendo se você criou um snapshot ou um backup:
   a. "Restaure seus dados a partir do snapshot".
   b. "Restaure seus dados a partir do backup".

Clone para cluster diferente

Para clonar um aplicativo para um cluster diferente (realizar uma clonagem entre clusters), crie um backup no cluster de origem e, em seguida, restaure o backup em um cluster diferente. Certifique-se de que o Trident Protect esteja instalado no cluster de destino.

É possível replicar um aplicativo entre clusters diferentes usando "Replicação SnapMirror"o .

Passos

1. "Crie uma cópia de segurança".
2. Verifique se o AppVault CR para o bucket de armazenamento de objetos que contém o backup foi configurado no cluster de destino.
3. No cluster de destino, "restaure seus dados a partir do backup".

Migrar aplicações de uma classe de storage para outra classe de storage

Você pode migrar aplicativos de uma classe de armazenamento para uma classe de armazenamento diferente restaurando um backup para a classe de armazenamento de destino.

Por exemplo (excluindo os segredos do CR de restauração):

```
apiVersion: protect.trident.netapp.io/v1
kind: SnapshotRestore
metadata:
  name: "${snapshotRestoreCRName}"
spec:
  appArchivePath: "${snapshotArchivePath}"
  appVaultRef: "${appVaultCRName}"
  namespaceMapping:
    - destination: "${destinationNamespace}"
      source: "${sourceNamespace}"
  storageClassMapping:
    - destination: "${destinationStorageClass}"
      source: "${sourceStorageClass}"
  resourceFilter:
    resourceMatchers:
      kind: Secret
      version: v1
    resourceSelectionCriteria: exclude
```

Restaure o instantâneo usando um CR

Passos

1. Crie o arquivo de recurso personalizado (CR) e nomeie-o `trident-protect-snapshot-restore-cr.yaml`.
2. No arquivo criado, configure os seguintes atributos:
   - **metadata.name**: (*required*) o nome deste recurso personalizado; escolha um nome único e sensível para o seu ambiente.
   - **Spec.appArchivePath**: O caminho dentro do AppVault onde o conteúdo do snapshot é armazenado. Você pode usar o seguinte comando para encontrar este caminho:

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

   - **Spec.appVaultRef**: (*required*) o nome do AppVault onde o conteúdo do instantâneo é armazenado.
   - **spec.namespaceMapping**: o mapeamento do namespace de origem da operação de restauração para o namespace de destino. Substitua `my-source-namespace` e `my-destination-namespace` por informações do seu ambiente.

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

3. Opcionalmente, se você precisar selecionar apenas certos recursos do aplicativo para restaurar, adicione filtragem que inclua ou exclua recursos marcados com rótulos específicos:
   - **ResourceFilter.resourceSelectionCriteria**: (Necessário para filtragem) Use `include or exclude` para incluir ou excluir um recurso definido em resourceMatchers. Adicione os seguintes parâmetros resourceMatchers para definir os recursos a serem incluídos ou excluídos:
     - **ResourceFilter.resourceMatchers**: Uma matriz de resourceMatcher objetos. Se você definir vários elementos nesse array, eles corresponderão como uma OPERAÇÃO OU, e os campos dentro de cada elemento (grupo, tipo, versão) corresponderão como uma OPERAÇÃO E.
       - **ResourceMatchers[].group**: (*Optional*) Grupo do recurso a ser filtrado.
       - **ResourceMatchers[].kind**: (*Opcional*) tipo do recurso a ser filtrado.
       - **ResourceMatchers[].version**: (*Optional*) versão do recurso a ser filtrado.

- **ResourceMatchers[].names**: (*Optional*) nomes no campo Kubernetes metadata.name do recurso a ser filtrado.
- **ResourceMatchers[].namespaces**: (*Optional*) namespaces no campo Kubernetes metadata.name do recurso a ser filtrado.
- **ResourceMatchers[].labelSelectors**: (*Optional*) string de seleção de etiquetas no campo Kubernetes metadata.name do recurso, conforme definido no "Documentação do Kubernetes". Por exemplo `"trident.netapp.io/os=linux":.`

  Por exemplo:

```
spec:
  resourceFilter:
    resourceSelectionCriteria: "include"
    resourceMatchers:
      - group: my-resource-group-1
        kind: my-resource-kind-1
        version: my-resource-version-1
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
      - group: my-resource-group-2
        kind: my-resource-kind-2
        version: my-resource-version-2
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
```

4. Depois de preencher o `trident-protect-snapshot-restore-cr.yaml` ficheiro com os valores corretos, aplique o CR:

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

Restaure o instantâneo usando a CLI

Passos

1. Restaure o snapshot para um namespace diferente, substituindo valores entre parênteses por informações do seu ambiente.

   - O `snapshot` argumento usa um namespace e um nome instantâneo no formato `<namespace>/<name>`.
   - O `namespace-mapping` argumento usa namespaces separados por dois pontos para mapear namespaces de origem para os namespaces de destino corretos no formato `source1:dest1,source2:dest2`.

     Por exemplo:

```
tridentctl-protect create snapshotrestore <my_restore_name>
--snapshot <namespace/snapshot_to_restore> --namespace-mapping
<source_to_destination_namespace_mapping>
```

Gerenciar os ganchos de execução do Trident Protect

Um gancho de execução é uma ação personalizada que você pode configurar para ser executada em conjunto com uma operação de proteção de dados de um aplicativo gerenciado. Por exemplo, se você tiver um aplicativo de banco de dados, poderá usar um gancho de execução para pausar todas as transações de banco de dados antes de um snapshot e retomar as transações após a conclusão do snapshot. Isso garante snapshots consistentes com aplicativos.

Tipos de ganchos de execução

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

- Pré-instantâneo
- Pós-snapshot
- Pré-backup
- Pós-backup
- Pós-restauração
- Pós-failover

Ordem de execução

Quando uma operação de proteção de dados é executada, os eventos de gancho de execução ocorrem na seguinte ordem:

1. Todos os ganchos de execução personalizados de pré-operação aplicáveis são executados nos contentores apropriados. Você pode criar e executar quantos ganchos de pré-operação personalizados você precisar, mas a ordem de execução desses ganchos antes da operação não é garantida nem configurável.
2. Congelamentos do sistema de arquivos ocorrem, se aplicável. "Saiba mais sobre como configurar o congelamento do sistema de arquivos com o Trident Protect.".
3. A operação de proteção de dados é realizada.
4. Os sistemas de arquivos congelados não estão congelados, se aplicável.
5. Todos os ganchos de execução pós-operação personalizados aplicáveis são executados nos contentores apropriados. Você pode criar e executar quantos ganchos de pós-operação personalizados você precisar, mas a ordem de execução desses ganchos após a operação não é garantida nem configurável.

Se você criar vários ganchos de execução do mesmo tipo (por exemplo, pré-snapshot), a ordem de execução desses ganchos não será garantida. No entanto, a ordem de execução de ganchos de diferentes tipos é garantida. Por exemplo, a seguinte é a ordem de execução de uma configuração que tem todos os diferentes

tipos de ganchos:

1. Ganchos pré-instantâneos executados
2. Ganchos pós-snapshot executados
3. Ganchos pré-backup executados
4. Ganchos pós-backup executados

> O exemplo de pedido 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 convenientemente os scripts. Depois de habilitar os ganchos de execução em um ambiente de produção, teste os snapshots e backups resultantes para garantir que eles sejam consistentes. Você pode fazer isso clonando o aplicativo para um namespace temporário, restaurando o snapshot ou o backup e testando o aplicativo.

> Se um gancho de execução pré-snapshot adicionar, alterar ou remover recursos do Kubernetes, essas alterações serão incluídas no snapshot ou backup e em qualquer operação de restauração subsequente.

Notas importantes sobre ganchos de execução personalizados

Considere o seguinte ao Planejar ganchos de execução para seus aplicativos.

- Um gancho de execução deve usar um script para executar ações. Muitos ganchos de execução podem referenciar o mesmo script.
- O Trident Protect exige que os scripts usados pelos ganchos de execução sejam escritos no formato de scripts shell executáveis.
- O tamanho do script está limitado a 96kbMB.
- O Trident Protect utiliza as configurações de ganchos de execução e quaisquer critérios correspondentes para determinar quais ganchos são aplicáveis a uma operação de snapshot, backup ou restauração.

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

Filtros de gancho de execução

Quando você adiciona ou edita um gancho de execução para um aplicativo, você pode adicionar filtros ao gancho de execução para gerenciar quais contentores o gancho corresponderá. Os filtros são úteis para aplicativos que usam a mesma imagem de contentor em todos os contentores, mas podem usar cada imagem para um propósito diferente (como o Elasticsearch). Os filtros permitem criar cenários onde os ganchos de execução são executados em alguns, mas não necessariamente em todos os contentores idênticos. Se você criar vários filtros para um único gancho de execução, eles serão combinados com um operador LÓGICO E. Você pode ter até 10 filtros ativos por gancho de execução.

Cada filtro que você adiciona a um gancho de execução usa uma expressão regular para corresponder aos contêineres no seu cluster. Quando um gancho corresponde a um contêiner, o gancho executará seu script associado naquele contêiner. Expressões regulares para filtros usam a sintaxe Regular Expression 2 (RE2), que não oferece suporte à criação de um filtro que exclua contêineres da lista de correspondências. Para obter informações sobre a sintaxe que o Trident Protect suporta para expressões regulares em filtros de gancho de execução, consulte "Suporte à sintaxe da expressão regular 2 (RE2)" .

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

Exemplos de gancho de execução

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

Crie um gancho de execução

Você pode criar um gancho de execução personalizado para um aplicativo usando o Trident Protect. Você precisa ter permissões de Proprietário, Administrador ou Membro para criar ganchos de execução.

Use um CR

Passos

1. Crie o arquivo de recurso personalizado (CR) e nomeie-o `trident-protect-hook.yaml`.
2. Configure os seguintes atributos para corresponder ao seu ambiente Trident Protect e à configuração do cluster:
   - **metadata.name**: (*required*) o nome deste recurso personalizado; escolha um nome único e sensível para o seu ambiente.
   - **Spec.applicationRef**: (*required*) o nome do Kubernetes do aplicativo para o qual executar o gancho de execução.
   - **Spec.stage**: (*required*) Uma cadeia de carateres indicando qual estágio durante a ação o gancho de execução deve ser executado. Valores possíveis:
     - Pre
     - Post
   - **Spec.action**: (*required*) Uma cadeia de carateres indicando qual ação o gancho de execução tomará, supondo que quaisquer filtros de gancho de execução especificados sejam correspondentes. Valores possíveis:
     - Snapshot
     - Backup
     - Restaurar
     - Failover
   - **Spec.enabled**: (*Optional*) indica se esse gancho de execução está ativado ou desativado. Se não for especificado, o valor padrão é verdadeiro.
   - **Spec.hookSource**: (*required*) Uma string contendo o script de gancho codificado em base64.
   - **Spec.timeout**: (*Optional*) Um número que define quanto tempo em minutos o gancho de execução pode ser executado. O valor mínimo é de 1 minuto e o valor padrão é de 25 minutos, se não for especificado.
   - **Spec.arguments**: (*Optional*) Uma lista YAML de argumentos que você pode especificar para o gancho de execução.
   - **Spec.matchingCriteria**: (*Optional*) uma lista opcional de pares de valores de chave de critérios, cada par compondo um filtro de gancho de execução. Você pode adicionar até 10 filtros por gancho de execução.
   - **Spec.matchingCriteria.type**: (*Optional*) Uma string que identifica o tipo de filtro do gancho de execução. Valores possíveis:
     - ContainerImage
     - Nome do ConteinerName
     - PodName
     - PodLabel
     - NamespaceName
   - **Spec.matchingCriteria.value**: (*Optional*) Uma string ou expressão regular identificando o valor do filtro do gancho de execução.

     Exemplo YAML:

```
apiVersion: protect.trident.netapp.io/v1
kind: ExecHook
metadata:
  name: example-hook-cr
  namespace: my-app-namespace
  annotations:
    astra.netapp.io/astra-control-hook-source-id:
/account/test/hookSource/id
spec:
  applicationRef: my-app-name
  stage: Pre
  action: Snapshot
  enabled: true
  hookSource: IyEvYmluL2Jhc2gKZWNobyAiZXhhbXBsZSBzY3JpcHQiCg==
  timeout: 10
  arguments:
    - FirstExampleArg
    - SecondExampleArg
  matchingCriteria:
    - type: containerName
      value: mysql
    - type: containerImage
      value: bitnami/mysql
    - type: podName
      value: mysql
    - type: namespaceName
      value: mysql-a
    - type: podLabel
      value: app.kubernetes.io/component=primary
    - type: podLabel
      value: helm.sh/chart=mysql-10.1.0
    - type: podLabel
      value: deployment-type=production
```

3. Depois de preencher o ficheiro CR com os valores corretos, aplique o CR:

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

Use a CLI

Passos

1. Crie o gancho de execução, substituindo valores entre parênteses por informações do seu ambiente. Por exemplo:

```
tridentctl-protect create exechook <my_exec_hook_name> --action
<action_type> --app <app_to_use_hook> --stage <pre_or_post_stage>
--source-file <script-file> -n <application_namespace>
```

Execute manualmente um gancho de execução

Você pode executar manualmente um gancho de execução para fins de teste ou se precisar executar novamente o gancho manualmente após uma falha. Você precisa ter permissões de proprietário, administrador ou membro para executar manualmente os ganchos de execução.

Executar manualmente um gancho de execução consiste em duas etapas básicas:

1. Crie um backup de recursos, que coleta recursos e cria um backup deles, determinando onde o gancho será executado
2. Execute o gancho de execução contra o backup

Passo 1: Crie um backup de recursos

Use um CR

Passos

1. Crie o arquivo de recurso personalizado (CR) e nomeie-o `trident-protect-resource-backup.yaml`.
2. Configure os seguintes atributos para corresponder ao seu ambiente Trident Protect e à configuração do cluster:
   - **metadata.name**: (*required*) o nome deste recurso personalizado; escolha um nome único e sensível para o seu ambiente.
   - **Spec.applicationRef**: (*required*) o nome do Kubernetes do aplicativo para o qual criar o backup de recursos.
   - **Spec.appVaultRef**: (*required*) o nome do AppVault onde o conteúdo de backup é armazenado.
   - **Spec.appArchivePath**: O caminho dentro do AppVault onde o conteúdo do backup é armazenado. Você pode usar o seguinte comando para encontrar este caminho:

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

     Exemplo YAML:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: ResourceBackup
metadata:
  name: example-resource-backup
spec:
  applicationRef: my-app-name
  appVaultRef: my-appvault-name
  appArchivePath: example-resource-backup
```

3. Depois de preencher o ficheiro CR com os valores corretos, aplique o CR:

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

Use a CLI

Passos

1. Crie o backup, substituindo valores entre parênteses por informações do seu ambiente. Por exemplo:

```
tridentctl protect create resourcebackup <my_backup_name> --app
<my_app_name> --appvault <my_appvault_name> -n
<my_app_namespace> --app-archive-path <app_archive_path>
```

2. Ver o estado da cópia de segurança. Você pode usar este comando de exemplo repetidamente até que a operação esteja concluída:

```
tridentctl protect get resourcebackup -n <my_app_namespace>
<my_backup_name>
```

3. Verifique se o backup foi bem-sucedido:

```
kubectl describe resourcebackup <my_backup_name>
```

Passo 2: Execute o gancho de execução

Use um CR

Passos

1. Crie o arquivo de recurso personalizado (CR) e nomeie-o `trident-protect-hook-run.yaml`.
2. Configure os seguintes atributos para corresponder ao seu ambiente Trident Protect e à configuração do cluster:
   - **metadata.name**: (*required*) o nome deste recurso personalizado; escolha um nome único e sensível para o seu ambiente.
   - **Spec.applicationRef**: (*required*) Certifique-se de que este valor corresponde ao nome da aplicação do ResourceBackup CR criado na etapa 1.
   - **Spec.appVaultRef**: (*required*) Certifique-se de que este valor corresponde ao appVaultRef do ResourceBackup CR criado na etapa 1.
   - **Spec.appArchivePath**: Certifique-se de que este valor corresponda ao appArchivePath do ResourceBackup CR criado na etapa 1.

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

   - **Spec.action**: (*required*) Uma cadeia de carateres indicando qual ação o gancho de execução tomará, supondo que quaisquer filtros de gancho de execução especificados sejam correspondentes. Valores possíveis:
     - Snapshot
     - Backup
     - Restaurar
     - Failover
   - **Spec.stage**: (*required*) Uma cadeia de carateres indicando qual estágio durante a ação o gancho de execução deve ser executado. Esta corrida de gancho não vai correr ganchos em qualquer outro estágio. Valores possíveis:
     - Pre
     - Post

     Exemplo YAML:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: ExecHooksRun
metadata:
  name: example-hook-run
spec:
  applicationRef: my-app-name
  appVaultRef: my-appvault-name
  appArchivePath: example-resource-backup
  stage: Post
  action: Failover
```

3. Depois de preencher o ficheiro CR com os valores corretos, aplique o CR:

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

Use a CLI

Passos

1. Crie a solicitação de execução manual do hook run:

```
tridentctl protect create exechooksrun <my_exec_hook_run_name>
-n <my_app_namespace> --action snapshot --stage <pre_or_post>
--app <my_app_name> --appvault <my_appvault_name> --path
<my_backup_name>
```

2. Verifique o status da execução do hook run. Você pode executar este comando repetidamente até que a operação esteja concluída:

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

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

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

Desinstale o Trident Protect.

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

Para remover o Trident Protect, siga os passos abaixo.

Passos

1. Remova os arquivos CR do Trident Protect:

   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