Gerenciar e proteger aplicativos

Trident

NetApp
September 26, 2025

Índice

Gerenciar e proteger aplicativos

Use objetos do Trident Protect AppVault para gerenciar buckets

O bucket custom resource (CR) do Trident Protect é conhecido como AppVault. Os objetos AppVault são a representação declarativa do fluxo de trabalho do Kubernetes de um bucket de storage. 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. Apenas os administradores podem criar AppVaults.

Geração de chaves e exemplos de definição do AppVault

Ao definir um AppVault CR, você precisa incluir credenciais para acessar os recursos hospedados pelo provedor. A forma como você gera as chaves para as credenciais será diferente dependendo do provedor. A seguir estão exemplos de geração de chaves de linha de comando para vários provedores, seguidos de exemplos de definições do AppVault para cada provedor.

Principais exemplos de geração

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=<generic-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=<generic-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
=<generic-s3-trident-protect-src-bucket-secret> -n trident-protect
```

Exemplos do AppVault CR

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

Google Cloud

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: gcp-trident-protect-src-bucket-b643cc50-0429-4ad5-971f-
ac4a83621922
  namespace: trident-protect
spec:
  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-b643cc50-0429-4ad5-971f-
ac4a83621922
  namespace: trident-protect
spec:
  providerType: AWS
  providerConfig:
    s3:
      bucketName: trident-protect-src-bucket
      endpoint: s3.example.com
  providerCredentials:
    accessKeyID:
      valueFromSecret:
        key: accessKeyID
        name: s3
    secretAccessKey:
      valueFromSecret:
        key: secretAccessKey
        name: s3
```

Microsoft Azure

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: azure-trident-protect-src-bucket-b643cc50-0429-4ad5-971f-
ac4a83621922
  namespace: trident-protect
spec:
  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-b643cc50-0429-4ad5-971f-
ac4a83621922
  namespace: trident-protect
spec:
  providerType: GenericS3
  providerConfig:
    s3:
      bucketName: trident-protect-src-bucket
      endpoint: s3.example.com
  providerCredentials:
    accessKeyID:
      valueFromSecret:
        key: accessKeyID
        name: s3
    secretAccessKey:
      valueFromSecret:
        key: secretAccessKey
        name: s3
```

ONTAP S3

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: ontap-s3-trident-protect-src-bucket-b643cc50-0429-4ad5-971f-
ac4a83621922
  namespace: trident-protect
spec:
  providerType: OntapS3
  providerConfig:
    s3:
      bucketName: trident-protect-src-bucket
      endpoint: s3.example.com
  providerCredentials:
    accessKeyID:
      valueFromSecret:
        key: accessKeyID
        name: s3
    secretAccessKey:
      valueFromSecret:
        key: secretAccessKey
        name: s3
```

StorageGRID S3

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: storagegrid-s3-trident-protect-src-bucket-b643cc50-0429-4ad5-
971f-ac4a83621922
  namespace: trident-protect
spec:
  providerType: StorageGridS3
  providerConfig:
    s3:
      bucketName: trident-protect-src-bucket
      endpoint: s3.example.com
  providerCredentials:
    accessKeyID:
      valueFromSecret:
        key: accessKeyID
        name: s3
    secretAccessKey:
      valueFromSecret:
        key: secretAccessKey
        name: s3
```

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.

Google Cloud

```
tridentctl-protect create vault GCP my-new-vault --bucket mybucket
--project my-gcp-project --secret <gcp-creds>/<credentials>
```

Amazon S3 (AWS)

```
tridentctl-protect create vault AWS <vault-name> --bucket <bucket-name>
--secret  <secret-name>  --endpoint <s3-endpoint>
```

Microsoft Azure

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

Genérico S3

```
tridentctl-protect create vault GenericS3 <vault-name> --bucket
<bucket-name> --secret  <secret-name>  --endpoint <s3-endpoint>
```

ONTAP S3

```
tridentctl-protect create vault OntapS3 <vault-name> --bucket <bucket-
name> --secret  <secret-name>  --endpoint <s3-endpoint>
```

StorageGRID S3

```
tridentctl-protect create vault StorageGridS3 s3vault --bucket <bucket-
name> --secret  <secret-name>  --endpoint <s3-endpoint>
```

Use o navegador AppVault para exibir informações do AppVault

Você pode usar o plugin Trident Protect CLI para exibir informações sobre objetos AppVault que foram criados no cluster.

Passos

1. Exibir o conteúdo de um objeto AppVault:

```
tridentctl-protect get appvaultcontent gcp-vault --show-resources all
```

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 leme Trident Protect. 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 snapshots e backups armazenados no bucket associado.

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 um aplicativo para gerenciamento com o Trident Protect

Você pode definir um aplicativo que deseja gerenciar com o Trident Protect criando um CR de aplicativo e um CR de 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 AppVault CR é específico para o seu ambiente; para exemplos do AppVault CRS, consulte "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 um CR de aplicativo ou usando a CLI 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 rótulos de namespace ou um nome de namespace para especificar namespaces nos quais os recursos da aplicação existem. O namespace da aplicação deve ser parte dessa lista.

        Exemplo YAML:

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

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

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

Adicione um aplicativo usando a CLI

Passos

1. Crie e aplique a definição da aplicação, substituindo valores entre parênteses por informações do seu ambiente. Você pode incluir namespaces e recursos na definição do aplicativo usando listas separadas por vírgulas com os argumentos mostrados 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>
```

Proteja aplicativos usando o Trident Protect

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

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.

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.

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**: (*Optional*): Nome do instantâneo a ser usado como fonte do backup. Se não for fornecido, um instantâneo temporário será criado e feito backup.

```
---
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> -n
<application_namespace>
```

Criar um cronograma de proteção de dados

Uma política de proteção protege um aplicativo criando snapshots, backups ou ambos em um cronograma definido. Você pode optar por criar snapshots e backups por hora, diariamente, semanalmente e mensalmente, e especificar o número de cópias a reter.

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**: O número de backups a reter. Zero indica que nenhum backup deve ser criado.
   - **Spec.snapshotRetention**: O número de instantâneos a reter. Zero indica que nenhum instantâneo deve ser criado.
   - **spec.granularity**: a frequência em que o horário deve ser executado. Valores possíveis, juntamente com campos associados obrigatórios:
     - `hourly` (requer que você `spec.minute` especifique)
     - `daily` (requer que você especifique `spec.minute` e `spec.hour`)
     - `weekly` (requer especificar `spec.minute, spec.hour`, e `spec.dayOfWeek`)
     - `monthly` (requer especificar `spec.minute, spec.hour`, e `spec.dayOfMonth`)
   - **Spec.dayOfMonth**: (*Optional*) o dia do mês (1 - 31) em que a programação deve ser executada. Este campo é necessário se a granularidade estiver definida como `monthly`.
   - **Spec.DayOfWeek**: (*Optional*) o dia da semana (0 - 7) em que o horário deve ser executado. Os valores de 0 ou 7 indicam domingo. Este campo é necessário se a granularidade estiver definida como `weekly`.
   - **Spec.hour**: (*Optional*) a hora do dia (0 - 23) em que o horário deve ser executado. Este campo é necessário se a granularidade estiver definida como `daily`, `weekly` `monthly` ou .
   - **Spec.minute**: (*Optional*) o minuto da hora (0 - 59) que o horário deve ser executado. Este campo é necessário se a granularidade estiver definida como `hourly`, , `daily` `weekly` , ou `monthly`.

```
---
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: <monthly>
  dayOfMonth: "1"
  dayOfWeek: "0"
  hour: "0"
  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> --data-mover
<kopia_or_restic> --day-of-month <day_of_month_to_run_schedule>
--day-of-week <day_of_month_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> -n <application_namespace>
```

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.

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 tiver instalado o Trident Protect, poderá habilitar a funcionalidade de backup e restauração com uso eficiente de espaço para back-ends de armazenamento que usam a classe de armazenamento azure-NetApp-Files e foram criados antes do Trident 24,06. Esta funcionalidade funciona com NFSv4 volumes 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. Esta aplicação terá uma funcionalidade de proteçãoão limitada até concluir 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 instantâneo não está ativado, 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 há espaço suficiente disponível no pool de capacidade para criar um volume temporário do tamanho do volume que está sendo feito backup.

Resultado

O aplicativo está pronto para backup e restauração usando o Trident Protect. Cada PVC também está disponível para ser usado por outras aplicações para backups e restaurações.

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 instantâneo existente será mais rápida ao restaurar o aplicativo para o 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.

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 usar o RedHat OpenShift, é importante observar o papel crítico das anotações de namespace em ambientes OpenShift. As anotações de namespace garantem que os pods restaurados aderem às permissões apropriadas e às configurações de segurança definidas pelas restrições de contexto de segurança OpenShift (SCCs) e possam acessar volumes sem problemas de permissão. Para obter 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:

```
kubectl set env -n trident-protect deploy/trident-protect-controller-
manager
RESTORE_SKIP_NAMESPACE_ANNOTATIONS=<annotation_key_to_skip_1>,<annotation_
key_to_skip_2>
```

Se instalou a aplicação de origem utilizando Helm com o `--create-namespace` sinalizador, é dado um tratamento especial à `name` tecla de identificação. 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

Restaure de um backup para um namespace diferente

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

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.
   - **Spec.storageClassMapping**: O mapeamento da classe de armazenamento de origem da operação de restauração para a classe de armazenamento de destino. Substitua `destinationStorageClass` e `sourceStorageClass` 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"}]
       storageClassMapping:
         destination: "${destinationStorageClass}"
         source: "${sourceStorageClass}"
     ```

3. (*Opcional*) se você precisar selecionar apenas determinados recursos do aplicativo para restaurar, 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":.`

  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.

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:
   - **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.

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.

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> -n <application_namespace>
--context <destination_cluster_name>
```

Restauração de um snapshot para um namespace diferente

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

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.
   - **Spec.storageClassMapping**: O mapeamento da classe de armazenamento de origem da operação de restauração para a classe de armazenamento de destino. Substitua `destinationStorageClass` e `sourceStorageClass` 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"}]
       storageClassMapping:
         destination: "${destinationStorageClass}"
         source: "${sourceStorageClass}"
     ```

3. (*Opcional*) se você precisar selecionar apenas determinados recursos do aplicativo para restaurar, 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"`: .

    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.

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:
   - **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 <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}'
```

Replique aplicações usando o NetApp SnapMirror e o Trident Protect

Com o Trident Protect, você pode usar os recursos de replicação assíncrona da tecnologia NetApp SnapMirror para replicar alterações de dados e aplicações de um back-end de storage 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 usar o RedHat OpenShift, é importante observar o papel crítico das anotações de namespace em ambientes OpenShift. As anotações de namespace garantem que os pods restaurados aderem às permissões apropriadas e às configurações de segurança definidas pelas restrições de contexto de segurança OpenShift (SCCs) e possam acessar volumes sem problemas de permissão. Para obter 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:

```
kubectl set env -n trident-protect deploy/trident-protect-controller-
manager
RESTORE_SKIP_NAMESPACE_ANNOTATIONS=<annotation_key_to_skip_1>,<annotation_
key_to_skip_2>
```

Se instalou a aplicação de origem utilizando Helm com o `--create-namespace` sinalizador, é dado um tratamento especial à `name` tecla de identificação. 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".

Configure uma relação de replicação

A configuração de uma relação de replicação envolve o seguinte:

- Escolhendo com que frequência você deseja que o Trident Protect tire um snapshot do aplicativo (que inclui os recursos do Kubernetes do aplicativo, bem como os snapshots de volume de 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. Crie um AppVault para o aplicativo de origem no cluster 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>
```

2. Criar 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, tire um snapshot do aplicativo de origem. Este instantâneo é utilizado como base para a aplicação no cluster de destino. Se você pular esta etapa, precisará esperar que o próximo snapshot agendado seja executado para que você tenha um snapshot recente.

Tire um instantâneo usando um CR

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**: (*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 da aplicação de origem CR.
- **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-0e1f88ab-f013-4bce-8ae9-6afed9df59a1
  namespace: my-app-namespace
spec:
  appVaultRef: generic-s3-trident-protect-src-bucket-04b6b4ec-
46a3-420a-b351-45795e1b5e34
  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
```

Tire um instantâneo usando a CLI

a. 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>
```

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

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

6. Crie um AppVault 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 my-app-namespace
```

7. Crie um arquivo CR AppMirrorRelationship:

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.storageClassName**: (*required*) escolha o nome de uma classe de armazenamento válida no cluster. A classe de storage deve ser vinculada a uma VM de storage do ONTAP que esteja vinculada ao ambiente de origem.
- **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 valores entre parênteses por informações do seu ambiente. Por exemplo:

```
tridentctl-protect create appmirrorrelationship <name_of_appmirorrelationship> --destination-app-vault <my_vault_name> --recurrence-rule <rule> --source-app <my_source_app> --source-app-vault <my_source_app_vault> -n <application_namespace>
```

8. (*Optional*) 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 fazer failover de aplicações replicadas para um cluster de destino. Este procedimento interrompe a relação de replicação e coloca a aplicação online no cluster de destino. O Trident Protect não interrompe o aplicativo no cluster de origem se ele estiver operacional.

Passos

1. Abra o arquivo CR do 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. Crie um instantâneo do aplicativo de origem.
2. Abra o arquivo CR do 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. Exclua o AppMirrorRelationship CR no cluster de destino original. 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 AppVault CRS esteja pronto em cada cluster.
4. Configure uma relação de replicação no cluster oposto, configurando valores para a direção inversa.

Sentido de replicação da aplicação inversa

Quando você inverte a direção da replicação, o Trident Protect move o aplicativo para o back-end de storage de destino e continua replicando de volta para o back-end de storage de origem original. O Trident Protect interrompe a aplicação de origem e replica os dados para o destino antes de fazer o failover para a aplicação de destino.

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

Passos

1. Criar 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. Após a conclusão do snapshot, obtenha o status do snapshot:

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

3. 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. Execute um failover do cluster de destino para o 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

Usando o Trident Protect, você pode obter "failback" após uma operação de failover usando a seguinte sequência de operações. Nesse fluxo de trabalho para restaurar a direção de replicação original, o Trident Protect replica (ressincrones) qualquer aplicativo é alterado de volta para o aplicativo de origem original antes de reverter a direção de 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. Excluir o AppMirrorRelationship CR:

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

Migrar aplicativos usando o Trident Protect

Você pode migrar seus aplicativos entre clusters ou classes de armazenamento restaurando seus dados de backup ou snapshot para um cluster ou classe de armazenamento diferente.

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 uma aplicação para um cluster diferente (executar um clone entre clusters), crie um backup no cluster de origem e restaure o backup para um cluster diferente. Certifique-se de que o Trident Protect está 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

É possível migrar aplicativos de uma classe de armazenamento para outra classe de armazenamento restaurando um snapshot para a classe de armazenamento de destino diferente.

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 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 eles 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. Ocorrem travamentos do sistema de arquivos, 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.

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 requer que os scripts que os ganchos de execução usam sejam escritos no formato de scripts shell executáveis.
- O tamanho do script está limitado a 96kbMB.
- O Trident Protect usa configurações de gancho 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 adicionar a um gancho de execução usa uma expressão regular para corresponder a containers em seu cluster. Quando um gancho corresponde a um recipiente, o gancho executará o script

associado nesse recipiente. As expressões regulares para filtros usam a sintaxe da expressão regular 2 (RE2), que não suporta a criação de um filtro que exclui contentores da lista de correspondências. Para obter informações sobre a sintaxe que o Trident Protect suporta para expressões regulares em filtros de gancho de execução, "Suporte à sintaxe da expressão regular 2 (RE2)" consulte .

> 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 ambiente do 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>
```

Informações sobre direitos autorais

Informações sobre marcas comerciais