Gerenciar e proteger aplicativos

Trident

NetApp
January 15, 2026

Índice

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.

Configure 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 Data Mover

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 solicitações de configuração (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 uma senha secreta para o repositório do Data Mover.

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 programa de transferência de dados que você estiver usando, basta incluir a senha correspondente a esse programa. Por exemplo, se você estiver usando o Restic e não planeja usar o Kopia no futuro, poderá incluir apenas a senha do Restic ao criar o segredo.
> - Guarde a senha em um local seguro. Você precisará dela para restaurar dados no mesmo cluster ou em um diferente. Se o cluster ou o `trident-protect` O namespace foi excluído e 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 IAM de armazenamento compatíveis 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 em "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 IAM de armazenamento compatíveis 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 a identidade do Pod do EKS, configure seu CR do AppVault 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
```

S3 genérico

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

ONTAP S3

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

StorageGrid S3

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

Exemplos de criação do AppVault

A seguir, apresentamos exemplos de definições do AppVault para cada provedor.

Exemplos de AppVault CR

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

- Opcionalmente, você pode especificar um segredo do Kubernetes que contenha senhas personalizadas para a criptografia dos repositórios Restic e Kopia. ConsulteSenhas do repositório do Data Mover para mais informações.
- Para objetos do Amazon S3 (AWS) AppVault, você pode opcionalmente especificar um sessionToken, o que é útil se você estiver usando o logon único (SSO) para autenticação. Este token é criado quando você gera chaves para o provedor emExemplos de geração de chaves AppVault para provedores de nuvem .
- Para objetos do AppVault no S3, você pode opcionalmente especificar um URL de proxy de saída para o tráfego de saída do S3 usando o `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 que utilizam o Pod Identity com o movimentador de dados Kopia, você pode remover o `providerCredentials` seção e adicione `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
```

S3 genérico

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

ONTAP S3

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

StorageGrid S3

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

Exemplos de criação do AppVault usando a CLI do Trident Protect

Você pode usar os seguintes exemplos de comandos da CLI para criar solicitações de configuração (CRs) do AppVault para cada provedor.

- Opcionalmente, você pode especificar um segredo do Kubernetes que contenha senhas personalizadas para a criptografia dos repositórios Restic e Kopia. ConsulteSenhas do repositório do Data Mover para mais informações.
- Para objetos do AppVault no S3, você pode opcionalmente especificar um URL de proxy de saída para o tráfego de saída do S3 usando o `--proxy-url <ip_address:port>` argumento.

Google Cloud

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

Amazon S3 (AWS)

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

Microsoft Azure

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

S3 genérico

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

ONTAP S3

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

StorageGrid S3

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

Veja as 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. Visualizar o conteúdo de um objeto do AppVault:

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

   Exemplo de saída:

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

2. Opcionalmente, para visualizar o AppVaultPath de cada recurso, use a flag `--show-paths` .

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

Remover um AppVault

Você pode remover um objeto do AppVault a qualquer momento.

> Não remova o `finalizers` Digite a chave no CR do AppVault antes de excluir o objeto do AppVault. Caso isso aconteça, poderá resultar em dados residuais no bucket do AppVault e recursos órfãos no cluster.

Antes de começar

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

Remova um AppVault usando a CLI do Kubernetes

1. Remova o objeto AppVault e substitua-o. `appvault-name` com o nome do objeto AppVault a ser removido:

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

Remova um AppVault usando a CLI do Trident Protect.

1. Remova o objeto AppVault e substitua-o. `appvault-name` com o nome do objeto AppVault a ser removido:

```
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.

Criar um AppVault CR

Você precisa criar um AppVault CR que será usado ao executar operações de proteção de dados no aplicativo, e o AppVault CR precisa residir no cluster onde 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.

Adicionar um aplicativo usando um CR

Passos

1. Crie o arquivo CR do aplicativo de destino:
   a. Crie o arquivo de recurso personalizado (CR) e dê um nome a ele (por exemplo, `maria-app.yaml`).
   b. Configure os seguintes atributos:
      - **metadata.name**: (*Obrigatório*) O nome do recurso personalizado do aplicativo. Anote o nome escolhido, pois outros arquivos CR necessários para operações de proteção farão referência a esse valor.
      - **spec.includedNamespaces**: (*Obrigatório*) Use o seletor de namespace e rótulo para especificar os namespaces e recursos que o aplicativo utiliza. O namespace da aplicação deve fazer parte desta lista. O seletor de rótulos é opcional e pode ser usado para filtrar recursos dentro de cada namespace especificado.
      - **spec.includedClusterScopedResources**: (Opcional) Use este atributo para especificar recursos com escopo de cluster a serem incluídos na definição do aplicativo. Este atributo permite selecionar esses recursos com base em seu grupo, versão, tipo e rótulos.
        - **groupVersionKind**: (*Obrigatório*) Especifica o grupo, a versão e o tipo da API do recurso com escopo de cluster.
        - **labelSelector**: (Opcional) Filtra os recursos com escopo de cluster com base em seus rótulos.
      - **metadata.annotations.protect.trident.netapp.io/skip-vm-freeze**: (*Opcional*) Esta anotação só se aplica a aplicações definidas a partir de máquinas virtuais, como em 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" .

Caso precise aplicar essa anotação após a criação de um aplicativo, 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 de 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 filtros que incluam ou excluam recursos marcados com rótulos específicos:

   - **resourceFilter.resourceSelectionCriteria**: (Obrigatório para filtragem) Use `Include` ou `Exclude` Para incluir ou excluir um recurso definido em resourceMatchers. Adicione os seguintes parâmetros resourceMatchers para definir os recursos a serem incluídos ou excluídos:

     - **resourceFilter.resourceMatchers**: Um array de objetos resourceMatcher. Se você definir vários elementos nessa matriz, eles corresponderão como uma operação OR, e os campos dentro de cada elemento (grupo, tipo, versão) corresponderão como uma operação AND.
       - **resourceMatchers[].group**: (Opcional) Grupo do recurso a ser filtrado.
       - **resourceMatchers[].kind**: (*Opcional*) Tipo do recurso a ser filtrado.
       - **resourceMatchers[].version**: (Opcional) Versão do recurso a ser filtrado.
       - **resourceMatchers[].names**: (Opcional) Nomes no campo metadata.name do Kubernetes do recurso a ser filtrado.

- **resourceMatchers[].namespaces**: (Opcional) Namespaces no campo metadata.name do recurso do Kubernetes a ser filtrado.
- **resourceMatchers[].labelSelectors**: (Opcional) String do seletor de rótulo no campo metadata.name do recurso do Kubernetes, conforme definido em "Documentação do Kubernetes" . Por exemplo: `"trident.netapp.io/os=linux"` .

> Quando ambos `resourceFilter` e `labelSelector` são usados, `resourceFilter` primeiro corre 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. Após criar a solicitação de configuração (CR) do aplicativo que corresponda ao seu ambiente, aplique a CR. Por exemplo:

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

Passos

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

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

captura de um snapshot. Se você usar a anotação, mas o aplicativo não tiver máquinas virtuais na definição do aplicativo, a anotação será ignorada. Se você não usar a anotação, o aplicativo seguirá o padrão."configuração de congelamento global do Trident Protect" .

Para especificar a anotação ao usar a CLI para criar um aplicativo, você pode usar o `--annotation` bandeira.

- Crie o aplicativo e utilize a configuração global para o comportamento de congelamento do sistema de arquivos:

```
tridentctl-protect create application <my_new_app_cr_name>
--namespaces <namespaces_to_include> --csr
<cluster_scoped_resources_to_include> --namespace <my-app-
namespace>
```

- Crie o aplicativo e configure as definições locais do aplicativo para o comportamento de congelamento do sistema de arquivos:

```
tridentctl-protect create application <my_new_app_cr_name>
--namespaces <namespaces_to_include> --csr
<cluster_scoped_resources_to_include> --namespace <my-app-
namespace> --annotation protect.trident.netapp.io/skip-vm-freeze
=<"true"|"false">
```

Você pode usar `--resource-filter-include` e `--resource-filter-exclude` sinalizadores para incluir ou excluir recursos com base em `resourceSelectionCriteria` tais 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.".

Criar um instantâneo sob demanda

Você pode criar um instantâneo 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.

Criar um instantâneo usando um CR

Passos

1. Crie o arquivo de recurso personalizado (CR) e dê um nome a ele. `trident-protect-snapshot-cr.yaml` .
2. No arquivo que você criou, configure os seguintes atributos:
   - **metadata.name**: (*Obrigatório*) O nome deste recurso personalizado; escolha um nome único e adequado ao seu ambiente.
   - **spec.applicationRef**: O nome do aplicativo no Kubernetes a ser criado a partir do snapshot.
   - **spec.appVaultRef**: (*Obrigatório*) O nome do AppVault onde o conteúdo do snapshot (metadados) deve ser armazenado.
   - **spec.reclaimPolicy**: (Opcional) Define o que acontece com o AppArchive de um snapshot quando o CR do snapshot é excluído. Isso significa que mesmo quando configurado para `Retain` , o instantâneo será excluído. Opções válidas:
     - `Retain`(padrão)
     - `Delete`

     ```
     ---
     apiVersion: protect.trident.netapp.io/v1
     kind: Snapshot
     metadata:
       namespace: my-app-namespace
       name: my-cr-name
     spec:
       applicationRef: my-application
       appVaultRef: appvault-name
       reclaimPolicy: Delete
     ```
3. Depois de preencher o `trident-protect-snapshot-cr.yaml` Arquivo 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 instantâneo, substituindo os valores entre colchetes pelas informações do seu ambiente. Por exemplo:

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

Crie um backup sob demanda

Você pode fazer backup de um 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 o tempo de expiração do token de sessão da AWS seja suficiente para quaisquer operações de backup do S3 de longa duração. Se o token expirar durante a operação de backup, a operação poderá falhar.

- Consulte o "Documentação do AWS API" 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 um backup usando um CR

Passos

1. Crie o arquivo de recurso personalizado (CR) e dê um nome a ele. `trident-protect-backup-cr.yaml` .
2. No arquivo que você criou, configure os seguintes atributos:
   - **metadata.name**: (*Obrigatório*) O nome deste recurso personalizado; escolha um nome único e adequado ao seu ambiente.
   - **spec.applicationRef**: (*Obrigatório*) O nome do aplicativo no Kubernetes para o qual deseja fazer backup.
   - **spec.appVaultRef**: (*Obrigatório*) O nome do AppVault onde o conteúdo do backup deve ser armazenado.
   - **spec.dataMover**: (Opcional) Uma string indicando qual ferramenta de backup usar para a operação de backup. Valores possíveis (diferencia maiúsculas de minúsculas):
     - `Restic`
     - `Kopia`(padrão)
   - **spec.reclaimPolicy**: (Opcional) Define o que acontece com um backup quando ele é liberado de sua reivindicação. Valores possíveis:
     - `Delete`
     - `Retain`(padrão)
   - **spec.snapshotRef**: (*Opcional*): Nome do snapshot a ser usado como fonte do backup. Caso não seja fornecida, será criada e armazenada uma cópia de segurança temporária.
   - **metadata.annotations.protect.trident.netapp.io/full-backup** : (*Opcional*) Esta anotação é usada para especificar se um backup deve ser não incremental. Por padrão, todos os backups são incrementais. No entanto, se esta anotação estiver definida como `true` , o backup deixa de ser incremental. Caso não seja especificado, o backup seguirá a configuração padrão de backup 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.

     Exemplo de YAML:

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

3. Depois de preencher o `trident-protect-backup-cr.yaml` Arquivo 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 os valores entre colchetes pelas informações do seu ambiente. Por exemplo:

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

   Você pode usar opcionalmente o `--full-backup` Sinalizador para especificar se um backup deve ser não incremental. Por padrão, todos os backups são incrementais. Quando essa opção é usada, o backup deixa de ser 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.

Crie 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 para snapshots somente configurando `backupRetention` para zero e `snapshotRetention` para um valor maior que zero. Contexto `snapshotRetention` Definir como zero significa que quaisquer backups agendados ainda criarão snapshots, mas estes serão temporários e excluídos imediatamente após a conclusão do backup.
- Os recursos com escopo de cluster são incluídos em um backup, snapshot ou clone se forem explicitamente referenciados na definição do aplicativo ou se tiverem referências a qualquer um dos namespaces do aplicativo.

Crie um cronograma usando uma CR (Central de Recursos).

Passos

1. Crie o arquivo de recurso personalizado (CR) e dê um nome a ele. `trident-protect-schedule-cr.yaml` .
2. No arquivo que você criou, configure os seguintes atributos:
   - **metadata.name**: (*Obrigatório*) O nome deste recurso personalizado; escolha um nome único e adequado ao seu ambiente.
   - **spec.dataMover**: (Opcional) Uma string indicando qual ferramenta de backup usar para a operação de backup. Valores possíveis (diferencia maiúsculas de minúsculas):
     - `Restic`
     - `Kopia`(padrão)
   - **spec.applicationRef**: O nome do aplicativo no Kubernetes para o qual será feito o backup.
   - **spec.appVaultRef**: (*Obrigatório*) O nome do AppVault onde o conteúdo do backup deve ser armazenado.
   - **spec.backupRetention**: O número de backups a serem retidos. Zero indica que nenhum backup deve ser criado (somente instantâneos).
   - **spec.snapshotRetention**: O número de snapshots a serem mantidos. Zero indica que nenhum instantâneo deve ser criado.
   - **spec.granularity**: A frequência com que a programação deve ser executada. Valores possíveis, juntamente com os campos associados obrigatórios:
     - `Hourly`(requer que você especifique) `spec.minute`)
     - `Daily`(requer que você especifique) `spec.minute` e `spec.hour`)
     - `Weekly`(requer que você especifique) `spec.minute, spec.hour` , e `spec.dayOfWeek`)
     - `Monthly`(requer que você especifique) `spec.minute, spec.hour` , e `spec.dayOfMonth`)
     - `Custom`
   - **spec.dayOfMonth**: (*Opcional*) O dia do mês (1 - 31) em que 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.
   - **metadata.annotations.protect.trident.netapp.io/full-backup-rule**: (Opcional) Esta anotação é usada para especificar a regra para agendamento de backup completo. 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 deverá ocorrer.

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
  annotations:
    protect.trident.netapp.io/full-backup-rule: "Monday,Thursday"
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` Arquivo com os valores corretos, aplique o CR:

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

Crie um agendamento usando a CLI (linha de comando)

Passos

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

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

   ```
   tridentctl-protect create schedule <my_schedule_name> --appvault
   <my_appvault_name> --app <name_of_app_to_snapshot> --backup
   -retention <how_many_backups_to_retain> --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>
   --full-backup-rule <string>
   ```

   Você pode configurar o `--full-backup-rule` bandeira 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 deverá ocorrer. Por exemplo, use `--full-backup-rule "Monday,Thursday"` Agendar backup completo às segundas e quintas-feiras.

   Para agendamentos somente de instantâneo, defina `--backup-retention 0` e especifique um valor maior que 0 para `--snapshot-retention` .

Excluir um instantâneo

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

Passos

1. Remova o CR (Código de Referência) do instantâneo associado ao instantâneo:

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

Excluir um backup

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

Certifique-se de que a política de reembolso esteja definida como `Delete` Remover todos os dados de backup do armazenamento de objetos. A configuração padrão da política é `Retain` Para evitar a perda acidental de dados. Se a política não for alterada para `Delete` Os dados de backup permanecerão no armazenamento de objetos e precisarão ser excluídos manualmente.

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 que está em andamento, foi concluída ou falhou.

Passos

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

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

Habilitar backup e restauração para 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

Garanta o seguinte:

- 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 tem `azure-netapp-files` Selecionada como a classe de armazenamento padrão para seu backend de armazenamento.

Expandir para ver as etapas de configuração

1. Se o volume ANF foi criado antes da atualização para o Trident 24.10, Trident os seguintes passos:

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

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

   b. Confirme se o diretório de snapshots foi habilitado 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 sistema 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.

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

Restaurar a partir de um backup para um namespace diferente.

Ao restaurar um backup para um namespace diferente usando um 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, mantendo os recursos existentes, não alterará nenhum recurso que compartilhe o mesmo nome que os recursos do backup. Para restaurar todos os recursos do backup, exclua e recrie o namespace de destino ou restaure o backup para um novo namespace.

Antes de começar

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

- Consulte o "Documentação do AWS API" 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 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 dê um nome a ele. `trident-protect-backup-restore-cr.yaml` .
2. No arquivo que você criou, configure os seguintes atributos:
   - **metadata.name**: (*Obrigatório*) O nome deste recurso personalizado; escolha um nome único e adequado ao seu ambiente.
   - **spec.appArchivePath**: O caminho dentro do AppVault onde o conteúdo do backup é armazenado. Você pode usar o seguinte comando para encontrar esse caminho:

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

   - **spec.appVaultRef**: (*Obrigatório*) O nome do AppVault onde o conteúdo do backup está armazenado.
   - **spec.namespaceMapping**: O mapeamento do namespace de origem da operação de restauração para o namespace de destino. Substituir `my-source-namespace` e `my-destination-namespace` com 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 precisar selecionar apenas determinados recursos do aplicativo para restaurar, adicione filtros que incluam ou excluam 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**: (Obrigatório para filtragem) Use `Include` ou `Exclude` Para incluir ou excluir um recurso definido em resourceMatchers. Adicione os seguintes parâmetros resourceMatchers para definir os recursos a serem incluídos ou excluídos:
     - **resourceFilter.resourceMatchers**: Um array de objetos resourceMatcher. Se você definir vários elementos nessa matriz, eles corresponderão como uma operação OR, e os campos

dentro de cada elemento (grupo, tipo, versão) corresponderão como uma operação AND.

- **resourceMatchers[].group**: (Opcional) Grupo do recurso a ser filtrado.
- **resourceMatchers[].kind**: (*Opcional*) Tipo do recurso a ser filtrado.
- **resourceMatchers[].version**: (Opcional) Versão do recurso a ser filtrado.
- **resourceMatchers[].names**: (Opcional) Nomes no campo metadata.name do Kubernetes do recurso a ser filtrado.
- **resourceMatchers[].namespaces**: (Opcional) Namespaces no campo metadata.name do recurso do Kubernetes a ser filtrado.
- **resourceMatchers[].labelSelectors**: (Opcional) String do seletor de rótulo no campo metadata.name do recurso do Kubernetes, conforme definido em "Documentação do Kubernetes" . Por exemplo: `"trident.netapp.io/os=linux"` .

  Por exemplo:

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

4. Depois de preencher o `trident-protect-backup-restore-cr.yaml` Arquivo 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 os valores entre colchetes pelas informações do seu ambiente. O `namespace-mapping` O 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>
```

Restaurar a partir 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 o tempo de expiração do token de sessão da AWS seja suficiente para quaisquer operações de restauração do S3 de longa duração. Se o token expirar durante a operação de restauração, a operação poderá falhar.

- Consulte o "Documentação do AWS API" 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 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 dê um nome a ele. `trident-protect-backup-ipr-cr.yaml` .
2. No arquivo que você criou, configure os seguintes atributos:
   - **metadata.name**: (*Obrigatório*) O nome deste recurso personalizado; escolha um nome único e adequado ao seu ambiente.
   - **spec.appArchivePath**: O caminho dentro do AppVault onde o conteúdo do backup é armazenado. Você pode usar o seguinte comando para encontrar esse caminho:

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

   - **spec.appVaultRef**: (*Obrigatório*) O nome do AppVault onde o conteúdo do backup está armazenado.

     Por exemplo:

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

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

   > 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**: (Obrigatório para filtragem) Use `Include` ou `Exclude` Para incluir ou excluir um recurso definido em resourceMatchers. Adicione os seguintes parâmetros resourceMatchers para definir os recursos a serem incluídos ou excluídos:
     - **resourceFilter.resourceMatchers**: Um array de objetos resourceMatcher. Se você definir vários elementos nessa matriz, eles corresponderão como uma operação OR, e os campos dentro de cada elemento (grupo, tipo, versão) corresponderão como uma operação AND.
       - **resourceMatchers[].group**: (Opcional) Grupo do recurso a ser filtrado.
       - **resourceMatchers[].kind**: (*Opcional*) Tipo do recurso a ser filtrado.

- **resourceMatchers[].version**: (Opcional) Versão do recurso a ser filtrado.
- **resourceMatchers[].names**: (Opcional) Nomes no campo metadata.name do Kubernetes do recurso a ser filtrado.
- **resourceMatchers[].namespaces**: (Opcional) Namespaces no campo metadata.name do recurso do Kubernetes a ser filtrado.
- **resourceMatchers[].labelSelectors**: (Opcional) String do seletor de rótulo no campo metadata.name do recurso do Kubernetes, conforme definido em "Documentação do Kubernetes" . Por exemplo: `"trident.netapp.io/os=linux"` .

  Por exemplo:

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

4. Depois de preencher o `trident-protect-backup-ipr-cr.yaml` Arquivo 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 os valores entre colchetes pelas informações do seu ambiente. O `backup` O argumento usa um namespace e um nome de backup no formato `<namespace>/<name>` . Por exemplo:

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

Restaurar a partir de um backup para um cluster diferente

Você pode restaurar um backup em um cluster diferente caso haja algum 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 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.

Antes de começar

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

- 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 está 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 o tempo de expiração do token de sessão da AWS seja suficiente para quaisquer operações de restauração de longa duração. Se o token expirar durante a operação de restauração, a operação poderá falhar.
  - Consulte o "Documentação do AWS API" 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>
```

> Certifique-se de que o namespace destinado à restauração do aplicativo exista no cluster de destino.

2. Visualize 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 este comando exibe os backups disponíveis no AppVault, incluindo seus clusters de origem, nomes de aplicativos correspondentes, registros de data e hora e caminhos de arquivamento.

Exemplo de saída:

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

3. Restaure o aplicativo no cluster de destino usando o nome do AppVault e o caminho do arquivo:

Use um CR

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

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

     Caso o BackupRestore CR não esteja disponível, você pode 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. Substituir `my-source-namespace` e `my-destination-namespace` com 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` Arquivo com os valores corretos, aplique o CR:

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

Use a CLI

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

dois pontos para mapear 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>
```

Restaurar a partir de um snapshot para um namespace diferente.

Você pode restaurar dados de um snapshot usando um arquivo de recurso personalizado (CR) para um namespace diferente ou para o namespace de origem original. Ao restaurar um snapshot para um namespace diferente usando um 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 é compatível com o `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` Se usar um sistema de armazenamento diferente, a operação de restauração falhará.

Antes de começar

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

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

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

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

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

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

   > 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**: (Obrigatório para filtragem) Use `Include` ou `Exclude` Para incluir ou excluir um recurso definido em resourceMatchers. Adicione os seguintes parâmetros resourceMatchers para definir os recursos a serem incluídos ou excluídos:
     - **resourceFilter.resourceMatchers**: Um array de objetos resourceMatcher. Se você definir vários elementos nessa matriz, eles corresponderão como uma operação OR, e os campos

dentro de cada elemento (grupo, tipo, versão) corresponderão como uma operação AND.

- **resourceMatchers[].group**: (Opcional) Grupo do recurso a ser filtrado.
- **resourceMatchers[].kind**: (*Opcional*) Tipo do recurso a ser filtrado.
- **resourceMatchers[].version**: (Opcional) Versão do recurso a ser filtrado.
- **resourceMatchers[].names**: (Opcional) Nomes no campo metadata.name do Kubernetes do recurso a ser filtrado.
- **resourceMatchers[].namespaces**: (Opcional) Namespaces no campo metadata.name do recurso do Kubernetes a ser filtrado.
- **resourceMatchers[].labelSelectors**: (Opcional) String do seletor de rótulo no campo metadata.name do recurso do Kubernetes, conforme definido em "Documentação do Kubernetes" . Por exemplo: `"trident.netapp.io/os=linux"` .

  Por exemplo:

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

4. Depois de preencher o `trident-protect-snapshot-restore-cr.yaml` Arquivo 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 os valores entre colchetes pelas informações do seu ambiente.

   - O `snapshot` O argumento usa um namespace e um nome de snapshot no formato `<namespace>/<name>` .
   - O `namespace-mapping` O 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>
```

Restaurar 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 o tempo de expiração do token de sessão da AWS seja suficiente para quaisquer operações de restauração do S3 de longa duração. Se o token expirar durante a operação de restauração, a operação poderá falhar.

- Consulte o "Documentação do AWS API" 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 dê um nome a ele. `trident-protect-snapshot-ipr-cr.yaml` .
2. No arquivo que você criou, configure os seguintes atributos:
   - **metadata.name**: (*Obrigatório*) O nome deste recurso personalizado; escolha um nome único e adequado ao seu ambiente.
   - **spec.appVaultRef**: (*Obrigatório*) O nome do AppVault onde o conteúdo do snapshot está armazenado.
   - **spec.appArchivePath**: O caminho dentro do AppVault onde o conteúdo do snapshot está armazenado. Você pode usar o seguinte comando para encontrar esse caminho:

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

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

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

   > 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**: (Obrigatório para filtragem) Use `Include` ou `Exclude` Para incluir ou excluir um recurso definido em resourceMatchers. Adicione os seguintes parâmetros resourceMatchers para definir os recursos a serem incluídos ou excluídos:
     - **resourceFilter.resourceMatchers**: Um array de objetos resourceMatcher. Se você definir vários elementos nessa matriz, eles corresponderão como uma operação OR, e os campos dentro de cada elemento (grupo, tipo, versão) corresponderão como uma operação AND.
       - **resourceMatchers[].group**: (Opcional) Grupo do recurso a ser filtrado.
       - **resourceMatchers[].kind**: (*Opcional*) Tipo do recurso a ser filtrado.
       - **resourceMatchers[].version**: (Opcional) Versão do recurso a ser filtrado.

- **resourceMatchers[].names**: (Opcional) Nomes no campo metadata.name do Kubernetes do recurso a ser filtrado.
- **resourceMatchers[].namespaces**: (Opcional) Namespaces no campo metadata.name do recurso do Kubernetes a ser filtrado.
- **resourceMatchers[].labelSelectors**: (Opcional) String do seletor de rótulo no campo metadata.name do recurso do Kubernetes, conforme definido em "Documentação do Kubernetes" . Por exemplo: `"trident.netapp.io/os=linux"` .

  Por exemplo:

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

4. Depois de preencher o `trident-protect-snapshot-ipr-cr.yaml` Arquivo 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 os valores entre colchetes pelas 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, foi concluída ou falhou.

Passos

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

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

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, os rótulos e anotações no namespace de destino são ajustados para corresponder aos rótulos e anotações no namespace de origem. Rótulos ou anotações do namespace de origem que não existem no namespace de destino são adicionados, e quaisquer rótulos ou anotações já existentes são sobrescritos para corresponder ao valor do namespace de origem. Rótulos ou anotações que existem apenas no espaço de nomes 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 sobre restrições de contexto de segurança do 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 --set
restoreSkipNamespaceAnnotations=<annotation_key_to_skip_1>,<annotation_key
_to_skip_2> --reuse-values
```

Ao executar uma operação de restauração ou failover, quaisquer anotações e rótulos de namespace especificados em `restoreSkipNamespaceAnnotations` e `restoreSkipNamespaceLabels` são excluídos da operação de restauração ou failover. Certifique-se de que essas configurações sejam definidas durante a instalação inicial do Helm. Para saber mais, consulte "Configurar opções de filtragem de namespace e AutoSupport".

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 um de destino, cada um com anotações e rótulos diferentes. É possível visualizar o estado do namespace de destino antes e depois da operação, bem como a forma como as anotações e os rótulos são combinados ou sobrescritos no namespace de destino.

Antes da operação de restauração ou failover

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

Espaço de nomes	Anotações	Etiquetas
Espaço de nomes ns-1 (fonte)	• anotação.um/chave: "valoratualizado" • anotação.dois/chave: "verdadeiro"	• ambiente=produção • conformidade=hipaa • nome=ns-1
Espaço de nomes ns-2 (destino)	• anotação.um/chave: "verdadeiro" • anotação.três/chave: "falso"	• função=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, outras foram sobrescritas, e a `name` O rótulo foi atualizado para corresponder ao namespace de destino:

Espaço de nomes	Anotações	Etiquetas
Espaço de nomes ns-2 (destino)	• anotação.um/chave: "valoratualizado" • anotação.dois/chave: "verdadeiro" • anotação.três/chave: "falso"	• nome=ns-2 • conformidade=hipaa • ambiente=produção • função=banco de dados

Campos suportados

Esta seção descreve campos adicionais disponíveis para operações de restauração.

Mapeamento de classes de armazenamento

O `spec.storageClassMapping` O 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 vários 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	corda	Tempo máximo (em segundos) permitido para a paralisação da operação de movimentação de dados.	"300"
protect.trident.netapp.io/kopia-content-cache-size-limit-mb	corda	O limite máximo de tamanho (em megabytes) para o cache de conteúdo do Kopia.	"1000"

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, os rótulos e anotações no namespace de destino são ajustados para corresponder aos rótulos e anotações no namespace de origem. Rótulos ou anotações do namespace de origem que não existem no namespace de destino são adicionados, e quaisquer rótulos ou anotações já existentes são sobrescritos para corresponder ao valor do namespace de origem. Rótulos ou anotações que existem apenas no espaço de nomes 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 sobre restrições de contexto de segurança do 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 --set
restoreSkipNamespaceAnnotations=<annotation_key_to_skip_1>,<annotation_key
_to_skip_2> --reuse-values
```

Ao executar uma operação de restauração ou failover, quaisquer anotações e rótulos de namespace especificados em `restoreSkipNamespaceAnnotations` e `restoreSkipNamespaceLabels` são excluídos da operação de restauração ou failover. Certifique-se de que essas configurações sejam definidas durante a instalação inicial do Helm. Para saber mais, consulte "Configurar opções de filtragem de namespace e AutoSupport".

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 um de destino, cada um com anotações e rótulos diferentes. É possível visualizar o estado do namespace de destino antes e depois da operação, bem como a forma como as anotações e os rótulos são combinados ou sobrescritos no namespace de destino.

Antes da operação de restauração ou failover

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

Espaço de nomes	Anotações	Etiquetas
Espaço de nomes ns-1 (fonte)	• anotação.um/chave: "valoratualizado" • anotação.dois/chave: "verdadeiro"	• ambiente=produção • conformidade=hipaa • nome=ns-1
Espaço de nomes ns-2 (destino)	• anotação.um/chave: "verdadeiro" • anotação.três/chave: "falso"	• função=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, outras foram sobrescritas, e a `name` O rótulo foi atualizado para corresponder ao namespace de destino:

Espaço de nomes	Anotações	Etiquetas
Espaço de nomes ns-2 (destino)	• anotação.um/chave: "valoratualizado" • anotação.dois/chave: "verdadeiro" • anotação.três/chave: "falso"	• nome=ns-2 • conformidade=hipaa • ambiente=produção • função=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 reversão

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

- Durante um failover, os hooks de execução são copiados automaticamente do cluster de origem para o cluster de destino. Você não precisa recriá-los manualmente. Após a falha, os ganchos de execução permanecem presentes na aplicação e serão executados durante quaisquer ações relevantes.
- Durante a reversão ou resincronização reversa, todos os ganchos de execução existentes no aplicativo são removidos. Quando a aplicação de origem se torna a aplicação de destino, esses ganchos de execução deixam de ser válidos e são excluídos para impedir sua execução.

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

Estabeleça uma relação de replicação.

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).
- Escolher o agendamento de replicação (inclui recursos do Kubernetes, bem como dados de volume persistentes)
- Definir a hora em que a foto será tirada.

Passos

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

Crie um AppVault usando um CR

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

b. Configure os seguintes atributos:

- **metadata.name**: (*Obrigatório*) O nome do recurso personalizado do AppVault. Anote o nome escolhido, pois outros arquivos CR necessários para uma relação de replicação farão referência a esse valor.
- **spec.providerConfig**: (*Obrigatório*) Armazena a configuração necessária para acessar o AppVault usando o provedor especificado. Escolha um nome para o bucket e quaisquer outros detalhes necessários para o seu provedor. Anote os valores que você escolher, pois outros arquivos CR necessários para uma relação de replicação fazem referência a esses valores. Consulte"Recursos personalizados do AppVault" Para exemplos de solicitações de configuração (CRs) do AppVault com outros provedores.
- **spec.providerCredentials**: (*Obrigatório*) Armazena referências a quaisquer credenciais necessárias para acessar o AppVault usando o provedor especificado.
  - **spec.providerCredentials.valueFromSecret**: (*Obrigatório*) Indica que o valor da credencial deve vir de um segredo.
    - **chave**: (*Obrigatório*) A chave válida do segredo a ser selecionado.
    - **nome**: (*Obrigatório*) Nome do segredo que contém o valor deste campo. Devem estar no mesmo espaço de nomes.
  - **spec.providerCredentials.secretAccessKey**: (*Obrigatório*) A chave de acesso usada para acessar o provedor. O **nome** deve corresponder a **spec.providerCredentials.valueFromSecret.name**.
- **spec.providerType**: (*Obrigatório*) Determina o que fornece o backup; por exemplo, NetApp ONTAP S3, S3 genérico, Google Cloud ou Microsoft Azure. Valores possíveis:
  - aws
  - azul
  - gcp
  - genérico-s3
  - ontap-s3
  - storagegrid-s3

c. Depois de preencher o `trident-protect-appvault-primary-source.yaml` Arquivo 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 os valores entre colchetes pelas 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 solicitação de configuração (CR) do aplicativo de origem:

Crie o aplicativo de origem usando um CR (Código de Referência).

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

b. Configure os seguintes atributos:

- **metadata.name**: (*Obrigatório*) O nome do recurso personalizado do aplicativo. Anote o nome escolhido, pois outros arquivos CR necessários para uma relação de replicação farão referência a esse valor.
- **spec.includedNamespaces**: (*Obrigatório*) Uma matriz de namespaces e rótulos associados. Utilize nomes de namespaces 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 deste array.

  Exemplo de YAML:

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

c. Depois de preencher o `trident-protect-app-source.yaml` Arquivo 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 (linha de comando)

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 "Criar um instantâneo sob demanda".

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

Além da programação fornecida abaixo, recomenda-se criar uma programação de snapshots diários separada, com um período de retenção de 7 dias, para manter um snapshot comum entre os clusters ONTAP interligados. Isso garante que os instantâneos fiquem disponíveis por até 7 dias, mas o período de retenção pode ser personalizado de acordo com as necessidades do usuário.

Caso ocorra uma falha, o sistema pode usar esses snapshots por até 7 dias para operações de reversão. Essa abordagem torna o processo inverso mais rápido e eficiente, pois apenas as alterações feitas desde o último instantâneo serão transferidas, e não todos os dados.

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

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 dê um nome a ele (por exemplo, `trident-protect-schedule.yaml`).

ii. Configure os seguintes atributos:

- **metadata.name**: (*Obrigatório*) O nome do recurso personalizado de agendamento.
- **spec.appVaultRef**: (*Obrigatório*) Este valor deve corresponder ao campo metadata.name do AppVault para o aplicativo de origem.
- **spec.applicationRef**: (*Obrigatório*) Este valor deve corresponder ao campo metadata.name do CR da aplicação de origem.
- **spec.backupRetention**: (*Obrigatório*) Este campo é obrigatório e o valor deve ser definido como 0.
- **spec.enabled**: Deve ser definido como verdadeiro.
- **spec.granularity**: Deve ser definido como `Custom` .
- **spec.recurrenceRule**: Define uma data de início em UTC e um intervalo de recorrência.
- **spec.snapshotRetention**: Deve ser definido como 2.

Exemplo de 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` Arquivo 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 AppVault CR de aplicativo de origem idêntico ao AppVault CR que você aplicou no cluster de origem e nomeie-o (por exemplo, `trident-protect-appvault-primary-destination.yaml`).
6. Aplicar o CR:

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

7. Crie uma solicitação de configuração (CR) do AppVault de destino para o aplicativo de destino no cluster de destino. Dependendo do seu provedor de armazenamento, modifique um exemplo em"Recursos personalizados do AppVault" para se adequar ao seu ambiente:
   a. Crie o arquivo de recurso personalizado (CR) e dê um nome a ele (por exemplo, `trident-protect-appvault-secondary-destination.yaml`).
   b. Configure os seguintes atributos:
      - **metadata.name**: (*Obrigatório*) O nome do recurso personalizado do AppVault. Anote o nome escolhido, pois outros arquivos CR necessários para uma relação de replicação farão referência a esse valor.
      - **spec.providerConfig**: (*Obrigatório*) Armazena a configuração necessária para acessar o AppVault usando o provedor especificado. Escolha um `bucketName` e quaisquer outros detalhes necessários para o seu fornecedor. Anote os valores que você escolher, pois outros arquivos CR necessários para uma relação de replicação fazem referência a esses valores. Consulte"Recursos personalizados do AppVault" Para exemplos de solicitações de configuração (CRs) do AppVault com outros provedores.
      - **spec.providerCredentials**: (*Obrigatório*) Armazena referências a quaisquer credenciais necessárias para acessar o AppVault usando o provedor especificado.

        - **spec.providerCredentials.valueFromSecret**: (*Obrigatório*) Indica que o valor da credencial deve vir de um segredo.
            - **chave**: (*Obrigatório*) A chave válida do segredo a ser selecionado.
            - **nome**: (*Obrigatório*) Nome do segredo que contém o valor deste campo. Devem estar no mesmo espaço de nomes.
        - **spec.providerCredentials.secretAccessKey**: (*Obrigatório*) A chave de acesso usada para acessar o provedor. O **nome** deve corresponder a **spec.providerCredentials.valueFromSecret.name**.
    - **spec.providerType**: (*Obrigatório*) Determina o que fornece o backup; por exemplo, NetApp ONTAP S3, S3 genérico, Google Cloud ou Microsoft Azure. Valores possíveis:
        - aws
        - azul
        - gcp
        - genérico-s3
        - ontap-s3
        - storagegrid-s3

c. Depois de preencher o `trident-protect-appvault-secondary-destination.yaml` Arquivo 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 de relacionamento AppMirror:

Crie um relacionamento AppMirror usando um CR

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

b. Configure os seguintes atributos:

- **metadata.name:** (Obrigatório) O nome do recurso personalizado AppMirrorRelationship.
- **spec.destinationAppVaultRef**: (*Obrigatório*) Este valor deve corresponder ao nome do AppVault para o aplicativo de destino no cluster de destino.
- **spec.namespaceMapping**: (*Obrigatório*) Os namespaces de destino e de origem devem corresponder ao namespace do aplicativo definido no respectivo CR do aplicativo.
- **spec.sourceAppVaultRef**: (*Obrigatório*) Este valor deve corresponder ao nome do AppVault para o aplicativo de origem.
- **spec.sourceApplicationName**: (*Obrigatório*) Este valor deve corresponder ao nome do aplicativo de origem que você definiu no CR do aplicativo de origem.
- **spec.sourceApplicationUID**: (Obrigatório) Este valor deve corresponder ao UID do aplicativo de origem que você definiu no CR do aplicativo de origem.
- **spec.storageClassName**: (Opcional) Escolha o nome de uma classe de armazenamento válida no cluster. A classe de armazenamento deve 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**: Define uma data de início em UTC e um intervalo de recorrência.

  Exemplo de 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` Arquivo com os valores corretos, aplique o CR:

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

Crie um relacionamento AppMirror 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. (Opcional) No cluster de destino, verifique o estado e o status da relação de replicação:

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

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. (Opcional) Crie quaisquer cronogramas de proteção que você precise no aplicativo em caso de falha.
5. (Opcional) Verifique o estado e o status da relação de replicação:

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

Ressincronizar uma relação de replicação que falhou

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

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

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

Passos

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

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

5. Se você criou algum agendamento de proteção no cluster de destino para proteger o aplicativo em failover, remova-o. Qualquer agendamento pendente causa falhas no snapshot de volume.

Reverter a resincronização de uma relação de replicação que falhou

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

Passos

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

Direção de replicação inversa do aplicativo

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.

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

Passos

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

Crie um instantâneo de desligamento usando uma CR (Código de Requisição).

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

b. Crie um arquivo CR de ShutdownSnapshot:

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

   ii. Configure os seguintes atributos:

   - **metadata.name**: (*Obrigatório*) O nome do recurso personalizado.
   - **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 arquivo CR do aplicativo de origem.

     Exemplo de 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` Arquivo com os valores corretos, aplique o CR:

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

Crie um instantâneo de desligamento usando a CLI

a. Crie o instantâneo de desligamento, substituindo os valores entre colchetes pelas informações do seu ambiente. Por exemplo:

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

2. No cluster de origem, após a conclusão do snapshot de desligamento, obtenha o status do snapshot de desligamento:

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

3. No cluster de origem, encontre o valor de **shutdownsnapshot.status.appArchivePath** usando o seguinte comando e registre a última parte do caminho do arquivo (também chamada de nome base; isso será tudo o que vem depois da última barra):

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

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

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

5. Execute as etapas de resincronização reversa emReverter a resincronização de uma relação de replicação que falhou .
6. Ative os agendamentos de proteção no novo cluster de origem.

Resultado

As seguintes ações ocorrem devido à replicação reversa:

- É feita uma captura instantânea dos recursos Kubernetes do aplicativo de origem original.
- Os pods do aplicativo de origem original são interrompidos corretamente excluindo os recursos do Kubernetes do aplicativo (mantendo os PVCs e PVs).
- Após o desligamento dos pods, são criados snapshots dos volumes do aplicativo e replicados.
- Os relacionamentos SnapMirror foram desfeitos, tornando os volumes de destino prontos para leitura/gravação.
- Os recursos do Kubernetes do aplicativo são restaurados a partir do snapshot anterior ao desligamento, usando os dados de volume replicados após o desligamento do aplicativo de origem original.
- A replicação é restabelecida na direção inversa.

Restaurar aplicativos para o cluster de origem 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 com um relacionamento que concluiu uma transição de failover para um destino e envolve as seguintes etapas:

- Comece com um estado de failover.
- Reverter a resincronização da relação de replicação.

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

- Inverta o sentido da replicação.

Passos

1. Realize oReverter a resincronização de uma relação de replicação que falhou passos.
2. Realize oDireção de replicação inversa do aplicativo passos.

Excluir uma relação de replicação

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

Passos

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

```
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.

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

Operações de backup e restauração

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

Clonar para o mesmo cluster

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

Passos

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

a. "Restaure seus dados a partir do snapshot.".

b. "Restaure seus dados a partir do backup.".

Clonar para um 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.

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

Passos

1. "Criar um backup".
2. Certifique-se de que 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 armazenamento para outra classe de armazenamento.

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
```

Restaurar o snapshot usando um CR

Passos

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

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

   - **spec.appVaultRef**: (*Obrigatório*) O nome do AppVault onde o conteúdo do snapshot está armazenado.
   - **spec.namespaceMapping**: O mapeamento do namespace de origem da operação de restauração para o namespace de destino. Substituir `my-source-namespace` e `my-destination-namespace` com informações do seu ambiente.

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

3. Opcionalmente, se precisar selecionar apenas determinados recursos do aplicativo para restaurar, adicione filtros que incluam ou excluam recursos marcados com rótulos específicos:
   - **resourceFilter.resourceSelectionCriteria**: (Obrigatório para filtragem) Use `include or exclude` Para incluir ou excluir um recurso definido em resourceMatchers. Adicione os seguintes parâmetros resourceMatchers para definir os recursos a serem incluídos ou excluídos:
     - **resourceFilter.resourceMatchers**: Um array de objetos resourceMatcher. Se você definir vários elementos nessa matriz, eles corresponderão como uma operação OR, e os campos dentro de cada elemento (grupo, tipo, versão) corresponderão como uma operação AND.
       - **resourceMatchers[].group**: (Opcional) Grupo do recurso a ser filtrado.
       - **resourceMatchers[].kind**: (*Opcional*) Tipo do recurso a ser filtrado.
       - **resourceMatchers[].version**: (Opcional) Versão do recurso a ser filtrado.

- **resourceMatchers[].names**: (Opcional) Nomes no campo metadata.name do Kubernetes do recurso a ser filtrado.
- **resourceMatchers[].namespaces**: (Opcional) Namespaces no campo metadata.name do recurso do Kubernetes a ser filtrado.
- **resourceMatchers[].labelSelectors**: (Opcional) String do seletor de rótulo no campo metadata.name do recurso do Kubernetes, conforme definido em "Documentação do Kubernetes" . Por exemplo: `"trident.netapp.io/os=linux"` .

  Por exemplo:

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

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

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

Restaure o snapshot usando a CLI

Passos

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

   - O `snapshot` O argumento usa um namespace e um nome de snapshot no formato `<namespace>/<name>` .
   - O `namespace-mapping` O 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 do banco de dados antes de um instantâneo e retomar as transações após a conclusão do instantâneo. Isso garante instantâneos consistentes com o aplicativo.

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-instantâneo
- 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 de pré-operação personalizados aplicáveis são executados nos contêineres apropriados. Você pode criar e executar quantos ganchos de pré-operação personalizados 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. Sistemas de arquivos congelados são descongelados, se aplicável.
5. Todos os ganchos de execução pós-operação personalizados aplicáveis são executados nos contêineres apropriados. Você pode criar e executar quantos ganchos pós-operação personalizados 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-instantâneo), a ordem de execução desses ganchos não será garantida. Entretanto, a ordem de execução de ganchos de diferentes tipos é garantida. Por exemplo, a seguir está a ordem de execução de uma configuração que possui todos os

diferentes tipos de ganchos:

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

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

Você deve sempre testar seus scripts de gancho de execução antes de habilitá-los em um ambiente de produção. Você pode usar o comando 'kubectl exec' para testar os scripts convenientemente. Depois de habilitar os ganchos de execução em um ambiente de produção, teste os snapshots e backups resultantes para garantir que sejam consistentes. Você pode fazer isso clonando o aplicativo em um namespace temporário, restaurando o snapshot ou backup e, em seguida, 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 é limitado a 96 KB.
- 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 desabilitam completamente a funcionalidade do aplicativo em que estão sendo executados, você deve sempre tentar minimizar o tempo que seus ganchos de execução personalizados levam 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

Ao adicionar ou editar um gancho de execução para um aplicativo, você pode adicionar filtros ao gancho de execução para gerenciar quais contêineres o gancho corresponderá. Os filtros são úteis para aplicativos que usam a mesma imagem de contêiner em todos os contêineres, mas podem usar cada imagem para uma finalidade diferente (como o Elasticsearch). Os filtros permitem que você crie cenários em que os ganchos de execução são executados em alguns contêineres idênticos, mas não necessariamente em todos. Se você criar vários filtros para um único gancho de execução, eles serão combinados com um operador lógico AND.

Você pode ter até 10 filtros ativos por gancho de execução.

Cada filtro que você adiciona a um gancho de execução usa uma expressão regular para corresponder 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 de Expressão Regular 2 (RE2)" .

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

Exemplos de ganchos 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 . 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 dê um nome a ele. `trident-protect-hook.yaml` .
2. Configure os seguintes atributos para corresponder ao seu ambiente Trident Protect e à configuração do cluster:
   - **metadata.name**: (*Obrigatório*) O nome deste recurso personalizado; escolha um nome único e adequado ao seu ambiente.
   - **spec.applicationRef**: (*Obrigatório*) O nome do aplicativo Kubernetes para o qual o gancho de execução será executado.
   - **spec.stage**: (*Obrigatório*) Uma string que indica em qual etapa da ação o gancho de execução deve ser executado. Valores possíveis:
     - Pré
     - Publicar
   - **spec.action**: (*Obrigatório*) Uma string que indica qual ação o gancho de execução tomará, assumindo que todos os filtros de gancho de execução especificados sejam correspondidos. Valores possíveis:
     - Instantâneo
     - Backup
     - Restaurar
     - Failover
   - **spec.enabled**: (Opcional) Indica se este gancho de execução está habilitado ou desabilitado. Caso não seja especificado, o valor padrão é verdadeiro.
   - **spec.hookSource**: (*Obrigatório*) Uma string contendo o script de gancho codificado em base64.
   - **spec.timeout**: (Opcional) Um número que define por quanto tempo, em minutos, o gancho de execução pode ser executado. O valor mínimo é de 1 minuto, e o valor padrão é de 25 minutos caso não seja especificado.
   - **spec.arguments**: (Opcional) Uma lista YAML de argumentos que você pode especificar para o gancho de execução.
   - **spec.matchingCriteria**: (*Opcional*) Uma lista opcional de pares de chave-valor de critérios, cada par constituindo um filtro de gancho de execução. Você pode adicionar até 10 filtros por gancho de execução.
   - **spec.matchingCriteria.type**: (*Opcional*) Uma string que identifica o tipo de filtro do gancho de execução. Valores possíveis:
     - Imagem do contêiner
     - Nome do contêiner
     - Nome do Pod
     - PodLabel
     - Nome do Namespace
   - **spec.matchingCriteria.value**: (*Opcional*) Uma string ou expressão regular que identifica o valor do filtro do gancho de execução.

Exemplo de YAML:

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

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

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

Use a CLI

Passos

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

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

Executar manualmente um gancho de execução

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

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

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

Passo 1: Crie um backup de recursos

Use um CR

Passos

1. Crie o arquivo de recurso personalizado (CR) e dê um nome a ele. `trident-protect-resource-backup.yaml` .
2. Configure os seguintes atributos para corresponder ao seu ambiente Trident Protect e à configuração do cluster:
   - **metadata.name**: (*Obrigatório*) O nome deste recurso personalizado; escolha um nome único e adequado ao seu ambiente.
   - **spec.applicationRef**: (*Obrigatório*) O nome do aplicativo Kubernetes para o qual o backup do recurso será criado.
   - **spec.appVaultRef**: (*Obrigatório*) O nome do AppVault onde o conteúdo do backup está armazenado.
   - **spec.appArchivePath**: O caminho dentro do AppVault onde o conteúdo do backup é armazenado. Você pode usar o seguinte comando para encontrar esse caminho:

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

     Exemplo de YAML:

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

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

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

Use a CLI

Passos

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

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

2. Veja o status do backup. Você pode usar este comando de exemplo repetidamente até que a operação seja concluída:

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

3. Verifique se o backup foi bem-sucedido:

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

Etapa 2: Execute o gancho de execução

Use um CR

Passos

1. Crie o arquivo de recurso personalizado (CR) e dê um nome a ele. `trident-protect-hook-run.yaml` .
2. Configure os seguintes atributos para corresponder ao seu ambiente Trident Protect e à configuração do cluster:
   - **metadata.name**: (*Obrigatório*) O nome deste recurso personalizado; escolha um nome único e adequado ao seu ambiente.
   - **spec.applicationRef**: (*Obrigatório*) Certifique-se de que este valor corresponda ao nome do aplicativo do ResourceBackup CR que você criou na etapa 1.
   - **spec.appVaultRef**: (*Obrigatório*) Certifique-se de que este valor corresponda ao appVaultRef do ResourceBackup CR que você criou na etapa 1.
   - **spec.appArchivePath**: Certifique-se de que esse valor corresponda ao appArchivePath do ResourceBackup CR que você criou na etapa 1.

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

   - **spec.action**: (*Obrigatório*) Uma string que indica qual ação o gancho de execução tomará, assumindo que todos os filtros de gancho de execução especificados sejam correspondidos. Valores possíveis:
     - Instantâneo
     - Backup
     - Restaurar
     - Failover
   - **spec.stage**: (*Obrigatório*) Uma string que indica em qual etapa da ação o gancho de execução deve ser executado. Esta sequência de ganchos não utilizará ganchos em nenhuma outra etapa. Valores possíveis:
     - Pré
     - Publicar

     Exemplo de YAML:

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

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

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

Use a CLI

Passos

1. Crie a solicitação de execução manual do gancho (hook):

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

2. Verifique o status da execução do gancho (hook). Você pode executar este comando repetidamente até que a operação seja concluída:

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

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

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

Informações sobre direitos autorais

Informações sobre marcas comerciais