



Protégez les applications avec Trident Protect

Trident

NetApp
September 26, 2025

Sommaire

Protégez les applications avec Trident Protect	1
Découvrez Trident Protect	1
Et la suite ?	1
Installez Trident Protect	1
Exigences de Trident Protect	1
Installer et configurer Trident Protect	4
Installez le plug-in Trident Protect CLI	10
Gérez Trident Protect	14
Gestion des autorisations et du contrôle d'accès Trident Protect	14
Générez un pack de support Trident Protect	20
Mettez à niveau Trident Protect	22
Gérez et protégez les applications	22
Utilisez les objets Trident Protect AppVault pour gérer les compartiments	22
Définissez une application de gestion avec Trident Protect	31
Protégez les applications à l'aide de Trident Protect	33
Restaurez les applications à l'aide de Trident Protect	41
Répliquez des applications à l'aide de NetApp SnapMirror et Trident Protect	57
Migrez vos applications à l'aide de Trident Protect	71
Gérer les crochets d'exécution de Trident Protect	75
Désinstallez Trident Protect	80

Protégez les applications avec Trident Protect

Découvrez Trident Protect

NetApp Trident Protect propose des fonctionnalités avancées de gestion des données applicatives qui améliorent la fonctionnalité et la disponibilité des applications Kubernetes avec état reposant sur les systèmes de stockage NetApp ONTAP et le mécanisme de provisionnement du stockage NetApp Trident CSI. Trident Protect simplifie la gestion, la protection et le déplacement des workloads conteneurisés dans les clouds publics et les environnements sur site. Il propose également des fonctionnalités d'automatisation via son API et son interface de ligne de commande.

Vous pouvez protéger les applications avec Trident Protect en créant des ressources personnalisées (CRS) ou en utilisant l'interface de ligne de commande Trident Protect.

Et la suite ?

Vous pouvez en savoir plus sur les exigences de Trident Protect avant de l'installer :

- ["Exigences de Trident Protect"](#)

Installez Trident Protect

Exigences de Trident Protect

Commencez par vérifier que votre environnement opérationnel, vos clusters d'applications, vos applications et vos licences sont prêts. Déploiement et exploitation de Trident Protect répondent à ces exigences.

Compatibilité des clusters Trident Protect Kubernetes

Trident Protect est compatible avec un large éventail d'offres Kubernetes entièrement gérées et autogérées, notamment :

- Amazon Elastic Kubernetes Service (EKS)
- Google Kubernetes Engine (GKE)
- Microsoft Azure Kubernetes Service (AKS)
- Red Hat OpenShift
- SUSE Rancher
- Gamme VMware Tanzu
- Kubernetes en amont



Assurez-vous que le cluster sur lequel vous installez Trident Protect est configuré avec un contrôleur de snapshot en cours d'exécution et les CRD associés. Pour installer un contrôleur de snapshot, reportez-vous à ["ces instructions"](#) la .

Trident protège la compatibilité back-end du stockage

Trident Protect prend en charge les systèmes back-end suivants :

- Amazon FSX pour NetApp ONTAP
- Cloud Volumes ONTAP
- Baies de stockage ONTAP
- Google Cloud NetApp volumes
- Azure NetApp Files

Assurez-vous que votre système back-end répond aux exigences suivantes :

- Assurez-vous que le stockage NetApp connecté au cluster utilise Astra Trident 24.02 ou version ultérieure (Trident 24.10 est recommandé).
 - Si Astra Trident est antérieure à la version 24.06.1 et que vous prévoyez d'utiliser la fonctionnalité de reprise d'activité NetApp SnapMirror, vous devez activer manuellement Astra Control provisionner.
- Vérifiez que vous disposez de la dernière version d'Astra Control Provisioner (installée et activée par défaut à partir d'Astra Trident 24.06.1).
- Assurez-vous de disposer d'un système back-end de stockage NetApp ONTAP.
- Assurez-vous d'avoir configuré un compartiment de stockage objet pour le stockage des sauvegardes.
- Créez tous les espaces de noms d'applications que vous prévoyez d'utiliser pour les opérations de gestion des données d'applications ou d'applications. Trident Protect ne crée pas ces espaces de noms pour vous ; si vous spécifiez un espace de noms inexistant dans une ressource personnalisée, l'opération échoue.

Conditions requises pour les volumes d'économie nas

Trident Protect prend en charge les opérations de sauvegarde et de restauration sur les volumes économiques nas. Les copies Snapshot, les clones et la réplication SnapMirror sur des volumes économiques nas ne sont pas pris en charge actuellement. Vous devez activer un répertoire d'instantanés pour chaque volume d'économie nas que vous prévoyez d'utiliser avec Trident Protect.



Certaines applications ne sont pas compatibles avec les volumes qui utilisent un répertoire de snapshots. Pour ces applications, vous devez masquer le répertoire des snapshots en exécutant la commande suivante sur le système de stockage ONTAP :

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

Vous pouvez activer le répertoire des snapshots en exécutant la commande suivante pour chaque volume nas-Economy, en remplaçant <volume-UUID> par l'UUID du volume à modifier :

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



Vous pouvez activer les répertoires de snapshots par défaut pour les nouveaux volumes en définissant l'option de configuration du back-end Trident `snapshotDir` sur `true`. Les volumes existants ne sont pas affectés.

Protéger les données avec les machines virtuelles KubeVirt

Trident Protect 24.10, 24.10.1 et versions ultérieures ont des comportements différents lorsque vous protégez les applications s'exécutant sur des machines virtuelles KubeVirt. Pour les deux versions, vous pouvez activer ou désactiver le blocage et le déblocage du système de fichiers pendant les opérations de protection des données.



Pour toutes les versions de Trident Protect, pour activer ou désactiver la fonctionnalité de blocage automatique dans les environnements OpenShift, vous devrez peut-être accorder des autorisations privilégiées à l'espace de noms d'applications. Par exemple :

```
oc adm policy add-scc-to-user privileged -z default -n  
<application-namespace>
```

Trident Protect 24.10

Trident Protect 24.10 ne garantit pas automatiquement un état cohérent pour les systèmes de fichiers des machines virtuelles KubeVirt lors des opérations de protection des données. Pour protéger les données de vos machines virtuelles KubeVirt à l'aide de Trident Protect 24.10, vous devez activer manuellement la fonctionnalité de gel/dégel pour les systèmes de fichiers avant l'opération de protection des données. Cela permet de s'assurer que les systèmes de fichiers sont dans un état cohérent.

Vous pouvez configurer Trident Protect 24.10 pour gérer le gel et le déblocage du système de fichiers de la machine virtuelle pendant les opérations de protection des données par "[configuration de la virtualisation](#)" puis à l'aide de la commande suivante :

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

Trident Protect 24.10.1 et versions ultérieures

Depuis Trident Protect 24.10.1, Trident Protect gèle et débloque automatiquement les systèmes de fichiers KubeVirt pendant les opérations de protection des données. Si vous le souhaitez, vous pouvez désactiver ce comportement automatique à l'aide de la commande suivante :

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

Conditions requises pour la réplication SnapMirror

NetApp SnapMirror est disponible pour les solutions ONTAP suivantes avec Trident Protect :

- NetApp ASA
- NetApp AFF
- NetApp FAS
- NetApp ONTAP Select
- NetApp Cloud Volumes ONTAP

- Amazon FSX pour NetApp ONTAP

Configuration requise pour un cluster ONTAP pour la réplication SnapMirror

Si vous prévoyez d'utiliser la réplication SnapMirror, assurez-vous que votre cluster ONTAP répond aux exigences suivantes :

- **Astra Control Provisioner ou Trident** : Astra Control Provisioner ou Trident doit exister sur les clusters Kubernetes source et de destination qui utilisent ONTAP en tant que back-end. Trident Protect prend en charge la réplication avec la technologie NetApp SnapMirror au moyen de classes de stockage basées sur les pilotes suivants :
 - `ontap-nas`
 - `ontap-san`
- **Licences** : les licences asynchrones de SnapMirror ONTAP utilisant le bundle protection des données doivent être activées sur les clusters ONTAP source et cible. Pour plus d'informations, reportez-vous à la section "[Présentation des licences SnapMirror dans ONTAP](#)".

Considérations de peering pour la réplication SnapMirror

Si vous prévoyez d'utiliser le peering back-end, assurez-vous que votre environnement répond aux exigences suivantes :

- **Cluster et SVM** : les systèmes back-end de stockage ONTAP doivent être peering. Pour plus d'informations, reportez-vous à la section "[Présentation du cluster et de SVM peering](#)".



S'assurer que les noms de SVM utilisés dans la relation de réplication entre deux clusters ONTAP sont uniques.

- **Astra Control Provisioner ou Trident et SVM** : les SVM distants à peering doivent être disponibles pour Astra Control Provisioner ou Trident sur le cluster destination.
- **Systèmes back-end gérés** : vous devez ajouter et gérer des systèmes back-end de stockage ONTAP dans Trident Protect pour créer une relation de réplication.
- **NVMe over TCP** : Trident Protect ne prend pas en charge la réplication NetApp SnapMirror pour les systèmes back-end de stockage qui utilisent le protocole NVMe over TCP.

Configuration Trident/ONTAP pour la réplication SnapMirror

Trident Protect exige que vous configuriez au moins un système back-end de stockage qui prend en charge la réplication à la fois pour les clusters source et de destination. Si les clusters source et cible sont identiques, l'application de destination doit utiliser un back-end de stockage différent de l'application source pour une résilience optimale.

Installer et configurer Trident Protect

Si votre environnement satisfait aux exigences de Trident Protect, vous pouvez suivre ces étapes pour installer Trident Protect sur votre cluster. Vous pouvez obtenir Trident Protect de NetApp ou l'installer à partir de votre propre registre privé. L'installation à partir d'un registre privé est utile si votre cluster ne peut pas accéder à Internet.



Par défaut, Trident Protect collecte des informations sur le support que vous pouvez ouvrir dans tous les dossiers de support NetApp, y compris les journaux, les metrics et les informations de topologie des clusters et des applications gérées. Trident Protect envoie ces offres de support à NetApp selon un calendrier quotidien. Vous pouvez éventuellement désactiver cette collection de packs de prise en charge lorsque vous installez Trident Protect. Vous pouvez le faire manuellement "[générer un bundle de support](#)" à tout moment.

Installez Trident Protect

Installez Trident Protect from NetApp

Étapes

1. Ajout du référentiel Trident Helm :

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

2. Installez les CRD Trident Protect :

```
helm install trident-protect-crds netapp-trident-protect/trident-  
protect-crds --version 100.2410.1 --create-namespace --namespace  
trident-protect
```

3. Utilisez Helm pour installer Trident Protect à l'aide de l'une des commandes suivantes. Remplacer `<name_of_cluster>` par un nom de cluster, qui sera attribué au cluster et utilisé pour identifier les sauvegardes et snapshots du cluster :

- Installez Trident Protect normalement :

```
helm install trident-protect netapp-trident-protect/trident-  
protect --set clusterName=<name_of_cluster> --version 100.2410.1  
--create-namespace --namespace trident-protect
```

- Installez Trident Protect et désactivez les téléchargements quotidiens de packs de prise en charge Trident Protect AutoSupport :

```
helm install trident-protect netapp-trident-protect/trident-  
protect --set autoSupport.enabled=false --set  
clusterName=<name_of_cluster> --version 100.2410.1 --create  
-namespace --namespace trident-protect
```

Installez Trident Protect à partir d'un registre privé

Vous pouvez installer Trident Protect à partir d'un registre d'images privé si votre cluster Kubernetes ne peut pas accéder à Internet. Dans ces exemples, remplacez les valeurs entre parenthèses par les informations de votre environnement :

Étapes

1. Extrayez les images suivantes sur votre ordinateur local, mettez à jour les balises, puis envoyez-les vers votre registre privé :


```
netapp/controller:24.10.1
netapp/restic:24.10.1
netapp/kopia:24.10.1
netapp/trident-autosupport:24.10.0
netapp/execheek:24.10.1
netapp/resourcebackup:24.10.1
netapp/resourcerestore:24.10.1
netapp/resourcedelete:24.10.1
bitnami/kubectl:1.30.2
kubebuilder/kube-rbac-proxy:v0.16.0
```

Par exemple :

```
docker pull netapp/controller:24.10.1
```

```
docker tag netapp/controller:24.10.1 <private-registry-
url>/controller:24.10.1
```

```
docker push <private-registry-url>/controller:24.10.1
```

2. Créer l'espace de noms du système Trident Protect :

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

3. Connectez-vous au registre :

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

4. Créez un secret Pull à utiliser pour l'authentification de registre privé :

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

5. Ajout du référentiel Trident Helm :

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

6. Créez un fichier nommé `protectValues.yaml`. Assurez-vous qu'il contient les paramètres Trident Protect suivants :

```
---
image:
  registry: <private-registry-url>
imagePullSecrets:
  - name: regcred
controller:
  image:
    registry: <private-registry-url>
rbacProxy:
  image:
    registry: <private-registry-url>
crCleanup:
  imagePullSecrets:
    - name: regcred
webhooksCleanup:
  imagePullSecrets:
    - name: regcred
```

7. Installez les CRD Trident Protect :

```
helm install trident-protect-crds netapp-trident-protect/trident-
protect-crds --version 100.2410.1 --create-namespace --namespace
trident-protect
```

8. Utilisez Helm pour installer Trident Protect à l'aide de l'une des commandes suivantes. Remplacer `<name_of_cluster>` par un nom de cluster, qui sera attribué au cluster et utilisé pour identifier les sauvegardes et snapshots du cluster :

- Installez Trident Protect normalement :

```
helm install trident-protect netapp-trident-protect/trident-
protect --set clusterName=<name_of_cluster> --version 100.2410.1
--create-namespace --namespace trident-protect -f
protectValues.yaml
```

- Installez Trident Protect et désactivez les téléchargements quotidiens de packs de prise en charge Trident Protect AutoSupport :

```
helm install trident-protect netapp-trident-protect/trident-protect --set autoSupport.enabled=false --set clusterName=<name_of_cluster> --version 100.2410.1 --create --namespace --namespace trident-protect -f protectValues.yaml
```

Spécifiez les limites des ressources du conteneur Trident Protect

Vous pouvez utiliser un fichier de configuration pour spécifier les limites de ressources pour les conteneurs Trident Protect après avoir installé Trident Protect. La définition de limites de ressources vous permet de contrôler la quantité de ressources du cluster consommées par les opérations Trident Protect.

Étapes

1. Créez un fichier nommé `resourceLimits.yaml`.
2. Remplissez le fichier avec les options de limite de ressources pour les conteneurs Trident Protect en fonction des besoins de votre environnement.

L'exemple de fichier de configuration suivant montre les paramètres disponibles et contient les valeurs par défaut pour chaque limite de ressource :

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

```

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

```

3. Appliquer les valeurs du `resourceLimits.yaml` fichier :

```

helm upgrade trident-protect -n trident-protect -f <resourceLimits.yaml>
--reuse-values

```

Installez le plug-in Trident Protect CLI

Vous pouvez utiliser le plug-in de ligne de commande Trident Protect, qui est une extension de l'utilitaire Trident `tridentctl`, pour créer et interagir avec les ressources personnalisées Trident Protect (CRS).

Installez le plug-in Trident Protect CLI

Avant d'utiliser l'utilitaire de ligne de commande, vous devez l'installer sur la machine que vous utilisez pour accéder à votre cluster. Procédez comme suit, selon si votre ordinateur utilise un processeur x64 ou ARM.

Télécharger le plug-in pour les processeurs Linux AMD64

Étapes

1. Téléchargez le plug-in Trident Protect CLI :

```
curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-protect/releases/download/24.10.1/tridentctl-protect-linux-amd64
```

Télécharger le plug-in pour les processeurs Linux ARM64

Étapes

1. Téléchargez le plug-in Trident Protect CLI :

```
curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-protect/releases/download/24.10.1/tridentctl-protect-linux-arm64
```

Télécharger le plug-in pour les processeurs Mac AMD64

Étapes

1. Téléchargez le plug-in Trident Protect CLI :

```
curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-protect/releases/download/24.10.1/tridentctl-protect-macos-amd64
```

Télécharger le plug-in pour les processeurs Mac ARM64

Étapes

1. Téléchargez le plug-in Trident Protect CLI :

```
curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-protect/releases/download/24.10.1/tridentctl-protect-macos-arm64
```

1. Activer les autorisations d'exécution pour le binaire du plug-in :

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

2. Copiez le fichier binaire du plug-in à un emplacement défini dans votre variable PATH. Par exemple, /usr/bin ou /usr/local/bin (vous pouvez avoir besoin d'un Privilèges élevé) :

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

3. Vous pouvez également copier le fichier binaire du plug-in vers un emplacement de votre répertoire personnel. Dans ce cas, il est recommandé de s'assurer que l'emplacement fait partie de votre variable PATH :

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



La copie du plug-in vers un emplacement de la variable PATH vous permet d'utiliser le plug-in en tapant `tridentctl-protect` ou `tridentctl protect` à partir de n'importe quel emplacement.

Afficher l'Trident aide du plug-in de l'interface de ligne

Vous pouvez utiliser les fonctions d'aide du plug-in intégré pour obtenir une aide détaillée sur les fonctionnalités du plug-in :

Étapes

1. Utilisez la fonction d'aide pour afficher les conseils d'utilisation :

```
tridentctl-protect help
```

Activer la saisie semi-automatique de la commande

Une fois que vous avez installé le plug-in de l'interface de ligne de commande Trident Protect, vous pouvez activer l'exécution automatique pour certaines commandes.

Activer la saisie semi-automatique pour le shell Bash

Étapes

1. Téléchargez le script d'achèvement :

```
curl -L -O https://github.com/NetApp/tridentctl-protect/releases/download/24.10.1/tridentctl-completion.bash
```

2. Créez un nouveau répertoire dans votre répertoire personnel pour contenir le script :

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

3. Déplacez le script téléchargé dans le ~/.bash/completions répertoire :

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

4. Ajoutez la ligne suivante au ~/.bashrc fichier de votre répertoire personnel :

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

Activer la saisie semi-automatique pour la coque Z.

Étapes

1. Téléchargez le script d'achèvement :

```
curl -L -O https://github.com/NetApp/tridentctl-protect/releases/download/24.10.1/tridentctl-completion.zsh
```

2. Créez un nouveau répertoire dans votre répertoire personnel pour contenir le script :

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

3. Déplacez le script téléchargé dans le ~/.zsh/completions répertoire :

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

4. Ajoutez la ligne suivante au ~/.zprofile fichier de votre répertoire personnel :

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

Résultat

Lors de votre prochaine connexion au shell, vous pouvez utiliser la saisie semi-automatique de la commande avec le plugin `tridentctl-protect`.

Gérez Trident Protect

Gestion des autorisations et du contrôle d'accès Trident Protect

Trident Protect utilise le modèle Kubernetes de contrôle d'accès basé sur des rôles (RBAC). Par défaut, Trident Protect fournit un espace de noms système unique et le compte de service par défaut qui lui est associé. Si vous avez une entreprise avec de nombreux utilisateurs ou des besoins de sécurité spécifiques, vous pouvez utiliser les fonctionnalités RBAC de Trident Protect pour bénéficier d'un contrôle plus granulaire sur l'accès aux ressources et aux espaces de noms.

L'administrateur du cluster a toujours accès aux ressources de l'espace de noms par défaut `trident-protect` et peut également accéder aux ressources de tous les autres espaces de noms. Pour contrôler l'accès aux ressources et aux applications, vous devez créer des espaces de noms supplémentaires et ajouter des ressources et des applications à ces espaces de noms.

Notez qu'aucun utilisateur ne peut créer de CRS de gestion des données d'application dans l'espace de noms par défaut `trident-protect`. Vous devez créer une CRS de gestion des données d'application dans un espace de noms d'application (pour cela, il est recommandé de créer une CRS de gestion des données d'application dans le même espace de nom que l'application associée).



Seuls les administrateurs doivent avoir accès aux objets de ressources personnalisés Privileged Trident Protect, notamment :

- **AppVault** : nécessite les données d'informations d'identification du compartiment
- **AutoSupportBundle** : collecte des mesures, des journaux et d'autres données sensibles de Trident Protect
- **AutoSupportBundleSchedule** : gère les plannings de collecte de journaux

Comme bonne pratique, utilisez RBAC pour limiter l'accès aux objets privilégiés aux administrateurs.

Pour plus d'informations sur la façon dont RBAC réglemente l'accès aux ressources et aux espaces de noms, reportez-vous à la section "[Documentation Kubernetes RBAC](#)".

Pour plus d'informations sur les comptes de service, reportez-vous au "[Documentation du compte de service Kubernetes](#)".

Exemple : gestion de l'accès pour deux groupes d'utilisateurs

Par exemple, une organisation dispose d'un administrateur de cluster, d'un groupe d'utilisateurs techniques et d'un groupe d'utilisateurs marketing. L'administrateur du cluster doit effectuer les tâches suivantes pour créer un environnement dans lequel le groupe d'ingénierie et le groupe marketing ont chacun accès uniquement aux ressources affectées à leurs espaces de noms respectifs.

Étape 1 : créez un espace de noms pour contenir des ressources pour chaque groupe

La création d'un espace de noms vous permet de séparer logiquement les ressources et de mieux contrôler qui a accès à ces ressources.

Étapes

1. Créer un espace de nom pour le groupe d'ingénierie :

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

2. Créez un espace de nom pour le groupe marketing :

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

Étape 2 : créez de nouveaux comptes de service pour interagir avec les ressources de chaque espace de noms

Chaque nouvel espace de noms que vous créez est fourni avec un compte de service par défaut, mais vous devez créer un compte de service pour chaque groupe d'utilisateurs afin de pouvoir diviser davantage Privileges entre les groupes si nécessaire.

Étapes

1. Créer un compte de service pour le groupe d'ingénierie :

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

2. Créez un compte de service pour le groupe marketing :

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

Étape 3 : créez un secret pour chaque nouveau compte de service

Un secret de compte de service est utilisé pour s'authentifier auprès du compte de service et peut facilement être supprimé et recréé si compromis.

Étapes

1. Créez un secret pour le compte de service d'ingénierie :

```

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

```

2. Créez un secret pour le compte de service marketing :

```

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

```

Étape 4 : créez un objet RoleBinding pour lier l'objet ClusterRole à chaque nouveau compte de service

Un objet ClusterRole par défaut est créé lorsque vous installez Trident Protect. Vous pouvez lier ce ClusterRole au compte de service en créant et en appliquant un objet RoleBinding.

Étapes

1. Liez ClusterRole au compte de service d'ingénierie :

```

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

```

2. Associez ClusterRole au compte de service marketing :

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

Étape 5 : autorisations de test

Vérifiez que les autorisations sont correctes.

Étapes

1. Vérifier que les utilisateurs d'ingénierie peuvent accéder aux ressources d'ingénierie :

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

2. Vérifiez que les utilisateurs d'ingénierie ne peuvent pas accéder aux ressources marketing :

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

Étape 6 : accorder l'accès aux objets AppVault

Pour effectuer des tâches de gestion des données telles que les sauvegardes et les snapshots, l'administrateur du cluster doit accorder l'accès aux objets AppVault à des utilisateurs individuels.

Étapes

1. Créez et appliquez un fichier YAML de combinaison AppVault et secret qui accorde à un utilisateur l'accès à un AppVault. Par exemple, la CR suivante accorde l'accès à un AppVault à l'utilisateur `eng-user`:

```

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

```

2. Créez et appliquez une CR de rôle pour permettre aux administrateurs de cluster d'accorder l'accès à des ressources spécifiques dans un espace de noms. Par exemple :

```

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

```

3. Créez et appliquez une CR RoleBinding pour lier les autorisations à l'utilisateur eng-user. Par exemple :

```

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

```

4. Vérifiez que les autorisations sont correctes.

a. Tentative de récupération des informations d'objet AppVault pour tous les espaces de noms :

```

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

```

Vous devez voir les résultats similaires à ce qui suit :

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

- b. Testez pour voir si l'utilisateur peut obtenir les informations AppVault qu'il a maintenant l'autorisation d'accéder :

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

Vous devez voir les résultats similaires à ce qui suit :

```
yes
```

Résultat

Les utilisateurs auxquels vous avez accordé des autorisations AppVault doivent pouvoir utiliser des objets AppVault autorisés pour les opérations de gestion des données applicatives et ne doivent pas pouvoir accéder à des ressources en dehors des espaces de noms attribués ou créer de nouvelles ressources auxquelles ils n'ont pas accès.

Générez un pack de support Trident Protect

Trident Protect permet aux administrateurs de générer des bundles qui incluent des informations utiles pour le support NetApp, notamment des journaux, des metrics et des informations de topologie sur les clusters et les applications à gérer. Si vous êtes connecté à Internet, vous pouvez télécharger des offres groupées de support sur le site de support NetApp (NSS) à l'aide d'un fichier de ressources personnalisées (CR).

Créez un bundle de support à l'aide d'une CR

Étapes

1. Créez le fichier de ressource personnalisée (CR) et nommez-le (par exemple, `trident-protect-support-bundle.yaml`).
2. Configurez les attributs suivants :
 - **metadata.name:** (*required*) le nom de cette ressource personnalisée; choisissez un nom unique et sensible pour votre environnement.
 - **Spec.triggerType:** (*required*) détermine si le bundle de support est généré immédiatement ou planifié. La génération planifiée du bundle a lieu à 12 h UTC. Valeurs possibles :
 - Planifié
 - Manuel
 - **Spec.uploadEnabled:** (*Optional*) détermine si le bundle de support doit être téléchargé sur le site de support NetApp après sa génération. Si ce n'est pas le cas, la valeur par défaut est `false`. Valeurs possibles :
 - vrai
 - false (valeur par défaut)
 - **Spec.dataWindowStart:** (*Optional*) chaîne de date au format RFC 3339 qui spécifie la date et l'heure auxquelles la fenêtre des données incluses dans le paquet de support doit commencer. Si ce n'est pas le cas, la valeur par défaut est de 24 heures. La date de fenêtre la plus ancienne que vous pouvez spécifier est il y a 7 jours.

Exemple YAML :

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

3. Une fois que vous avez rempli le `astra-support-bundle.yaml` fichier avec les valeurs correctes, appliquez la CR :

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

Créez un bundle de support à l'aide de l'interface de ligne de commande

Étapes

1. Créez le pack de support en remplaçant les valeurs entre parenthèses par les informations de votre environnement. `trigger-type` Détermine si le bundle est créé immédiatement ou

si l'heure de création est déterminée par le planning, et peut être `Manual` ou `Scheduled`. Le paramètre par défaut est `Manual`.

Par exemple :

```
tridentctl-protect create autosupportbundle <my_bundle_name>  
--trigger-type <trigger_type>
```

Mettez à niveau Trident Protect

Vous pouvez mettre à niveau Trident Protect vers la dernière version pour bénéficier de nouvelles fonctionnalités ou de nouveaux correctifs.

Pour mettre à niveau Trident Protect, procédez comme suit.

Étapes

1. Mettez à jour le référentiel Trident Helm :

```
helm repo update
```

2. Mettez à niveau les CRD Trident Protect :

```
helm upgrade trident-protect-crds netapp-trident-protect/trident-  
protect-crds --version 100.2410.1 --namespace trident-protect
```

3. Mise à niveau de Trident Protect :

```
helm upgrade trident-protect netapp-trident-protect/trident-protect  
--version 100.2410.1 --namespace trident-protect
```

Gérez et protégez les applications

Utilisez les objets Trident Protect AppVault pour gérer les compartiments

La ressource personnalisée de compartiment (CR) pour Trident Protect est appelée AppVault. Les objets AppVault sont la représentation déclarative du workflow Kubernetes d'un compartiment de stockage. Une CR AppVault contient les configurations nécessaires à l'utilisation d'un compartiment dans les opérations de protection, telles que les sauvegardes, les snapshots, les opérations de restauration et la réplication SnapMirror. Seuls les administrateurs peuvent créer des AppVault.

Exemples de génération de clés et de définition d'AppVault

Lors de la définition d'une CR AppVault, vous devez inclure des informations d'identification pour accéder aux ressources hébergées par le fournisseur. La façon dont vous générez les clés pour les informations d'identification varie en fonction du fournisseur. Vous trouverez ci-dessous des exemples de génération de clés de ligne de commande pour plusieurs fournisseurs, suivis des exemples de définitions AppVault pour chaque fournisseur.

Exemples de génération de clés

Vous pouvez utiliser les exemples suivants pour créer des clés pour les identifiants de chaque fournisseur cloud.

Google Cloud

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

Amazon S3 (AWS)

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

Microsoft Azure

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

S3 générique

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

ONTAP S3

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

StorageGRID S3

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

Exemples de CR AppVault

Vous pouvez utiliser les exemples CR suivants pour créer des objets AppVault pour chaque fournisseur de cloud.

Google Cloud

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: gcp-trident-protect-src-bucket-b643cc50-0429-4ad5-971f-ac4a83621922
  namespace: trident-protect
spec:
  providerType: GCP
  providerConfig:
    gcp:
      bucketName: trident-protect-src-bucket
      projectID: project-id
  providerCredentials:
    credentials:
      valueFromSecret:
        key: credentials
        name: gcp-trident-protect-src-bucket-secret
```

Amazon S3 (AWS)

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

Microsoft Azure

```

apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: azure-trident-protect-src-bucket-b643cc50-0429-4ad5-971f-
ac4a83621922
  namespace: trident-protect
spec:
  providerType: Azure
  providerConfig:
    azure:
      accountName: account-name
      bucketName: trident-protect-src-bucket
  providerCredentials:
    accountKey:
      valueFromSecret:
        key: accountKey
        name: azure-trident-protect-src-bucket-secret

```

S3 générique

```

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

```

ONTAP S3

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

StorageGRID S3

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

Exemples de création d'AppVault à l'aide de l'interface de ligne de commande Trident Protect

Vous pouvez utiliser les exemples de commandes CLI suivants pour créer AppVault CRS pour chaque fournisseur.

Google Cloud

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

Amazon S3 (AWS)

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

Microsoft Azure

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

S3 générique

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

ONTAP S3

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

StorageGRID S3

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

Utilisez le navigateur AppVault pour afficher les informations AppVault

Vous pouvez utiliser le plug-in Trident Protect CLI pour afficher des informations sur les objets AppVault qui ont été créés sur le cluster.

Étapes

1. Afficher le contenu d'un objet AppVault :

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

Exemple de sortie :

```

+-----+-----+-----+-----+
+-----+
|  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. Si vous le souhaitez, utilisez l'indicateur pour afficher le chemin d'accès à l'application pour chaque ressource `--show-paths`.

Le nom de cluster figurant dans la première colonne du tableau n'est disponible que si un nom de cluster a été spécifié dans l'installation de Trident Protect Helm. Par exemple : `--set clusterName=production1`.

Supprimer un AppVault

Vous pouvez supprimer un objet AppVault à tout moment.



Ne supprimez pas la `finalizers` clé dans la CR AppVault avant de supprimer l'objet AppVault. Dans ce cas, des données résiduelles dans le compartiment AppVault et des ressources orphelines dans le cluster.

Avant de commencer

Assurez-vous d'avoir supprimé tous les snapshots et les sauvegardes stockés dans le compartiment associé.

Supprimez un AppVault à l'aide de l'interface de ligne de commande Kubernetes

1. Supprimez l'objet AppVault, en le remplaçant `appvault_name` par le nom de l'objet AppVault à supprimer :

```
kubectl delete appvault <appvault_name> -n trident-protect
```

Supprimez un AppVault à l'aide de l'interface de ligne de commande Trident Protect

1. Supprimez l'objet AppVault, en le remplaçant `appvault_name` par le nom de l'objet AppVault à supprimer :

```
tridentctl-protect delete appvault <appvault_name> -n trident-protect
```

Définissez une application de gestion avec Trident Protect

Vous pouvez définir une application que vous souhaitez gérer avec Trident Protect en créant une application CR et une application CR AppVault associée.

Créez une CR AppVault

Vous devez créer une CR AppVault qui sera utilisée lors des opérations de protection des données sur l'application et la CR AppVault doit résider sur le cluster sur lequel Trident Protect est installé. La CR AppVault est spécifique à votre environnement ; pour obtenir des exemples de CRS AppVault, reportez-vous à la section ["Ressources personnalisées AppVault."](#)

Définir une application

Vous devez définir chaque application à gérer avec Trident Protect. Vous pouvez définir une application à gérer en créant manuellement une application CR ou en utilisant l'interface de ligne de commande Trident Protect.

Ajouter une application à l'aide d'une demande de modification

Étapes

1. Créez le fichier CR de l'application de destination :
 - a. Créez le fichier de ressource personnalisée (CR) et nommez-le (par exemple, `maria-app.yaml`).
 - b. Configurez les attributs suivants :
 - **metadata.name:** (*required*) le nom de la ressource personnalisée de l'application. Notez le nom que vous choisissez car les autres fichiers CR nécessaires aux opérations de protection font référence à cette valeur.
 - **spec.includedNamespaces:** (*required*) utilisez des étiquettes d'espace de noms ou un nom d'espace de noms pour spécifier les espaces de noms dans lesquels les ressources d'application existent. L'espace de noms de l'application doit faire partie de cette liste.

Exemple YAML :

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

2. Une fois que vous avez créé la demande de modification de l'application pour l'adapter à votre environnement, appliquez la demande de modification. Par exemple :

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

Ajoutez une application à l'aide de l'interface de ligne de commande

Étapes

1. Créez et appliquez la définition de l'application, en remplaçant les valeurs entre parenthèses par les informations de votre environnement. Vous pouvez inclure des espaces de noms et des ressources dans la définition d'application à l'aide de listes séparées par des virgules avec les arguments indiqués dans l'exemple suivant :

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

Protégez les applications à l'aide de Trident Protect

Vous pouvez protéger toutes les applications gérées par Trident Protect en effectuant des snapshots et des sauvegardes à l'aide d'une règle de protection automatisée ou ad hoc.



Vous pouvez configurer Trident Protect pour qu'il bloque et dégèle les systèmes de fichiers pendant les opérations de protection des données. ["En savoir plus sur la configuration de la congélation du système de fichiers avec Trident Protect"](#).

Créer un snapshot à la demande

Vous pouvez créer un snapshot à la demande à tout moment.

Créer un instantané à l'aide d'une CR

Étapes

1. Créez le fichier de ressource personnalisée (CR) et nommez-le `trident-protect-snapshot-cr.yaml`.
2. Dans le fichier que vous avez créé, configurez les attributs suivants :
 - **metadata.name**: (*required*) le nom de cette ressource personnalisée; choisissez un nom unique et sensible pour votre environnement.
 - **Spec.applicationRef** : nom Kubernetes de l'application à snapshot.
 - **Spec.appVaultRef**: (*required*) Nom de l'AppVault où le contenu de l'instantané (métadonnées) doit être stocké.
 - **Spec.reclaimPolicy**: (*Optional*) définit ce qui arrive à l'AppArchive d'un snapshot lorsque le snapshot CR est supprimé. Cela signifie que même si la valeur est définie sur Retain, l'instantané sera supprimé. Options valides :
 - Retain (par défaut)
 - 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. Une fois que vous avez rempli le `trident-protect-snapshot-cr.yaml` fichier avec les valeurs correctes, appliquez la CR :

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

Créer un snapshot à l'aide de l'interface de ligne de commandes

Étapes

1. Créez l'instantané, en remplaçant les valeurs entre parenthèses par les informations de votre environnement. Par exemple :

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

Créez une sauvegarde à la demande

Vous pouvez sauvegarder une application à tout moment.

Créez une sauvegarde à l'aide d'une CR

Étapes

1. Créez le fichier de ressource personnalisée (CR) et nommez-le `trident-protect-backup-cr.yaml`.
2. Dans le fichier que vous avez créé, configurez les attributs suivants :
 - **metadata.name:** (*required*) le nom de cette ressource personnalisée; choisissez un nom unique et sensible pour votre environnement.
 - **Spec.applicationRef:** (*required*) Nom Kubernetes de l'application à sauvegarder.
 - **Spec.appVaultRef:** (*required*) Nom de l'AppVault où le contenu de la sauvegarde doit être stocké.
 - **Spec.datamover:** (*Optional*) chaîne indiquant l'outil de sauvegarde à utiliser pour l'opération de sauvegarde. Valeurs possibles (sensibles à la casse) :
 - Restic
 - Kopia (par défaut)
 - **Spec.reclaimPolicy:** (*Optional*) définit ce qui arrive à une sauvegarde lorsqu'elle est libérée de sa réclamation. Valeurs possibles :
 - Delete
 - Retain (par défaut)
 - **Spec.snapshotRef:** (*Optional*): Nom du snapshot à utiliser comme source de la sauvegarde. Si ce n'est pas le cas, un instantané temporaire sera créé et sauvegardé.

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

3. Une fois que vous avez rempli le `trident-protect-backup-cr.yaml` fichier avec les valeurs correctes, appliquez la CR :

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

Créez une sauvegarde à l'aide de l'interface de ligne de commande

Étapes

1. Créez la sauvegarde en remplaçant les valeurs entre parenthèses par les informations de votre environnement. Par exemple :

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

Créez un calendrier de protection des données

Une règle de protection protège une application en créant des snapshots, des sauvegardes ou les deux à un calendrier défini. Vous pouvez choisir de créer des snapshots et des sauvegardes toutes les heures, tous les jours, toutes les semaines et tous les mois, et vous pouvez spécifier le nombre de copies à conserver.

Créer un programme à l'aide d'une demande de modification

Étapes

1. Créez le fichier de ressource personnalisée (CR) et nommez-le `trident-protect-schedule-cr.yaml`.
2. Dans le fichier que vous avez créé, configurez les attributs suivants :
 - **metadata.name:** (*required*) le nom de cette ressource personnalisée; choisissez un nom unique et sensible pour votre environnement.
 - **Spec.datamover:** (*Optional*) chaîne indiquant l'outil de sauvegarde à utiliser pour l'opération de sauvegarde. Valeurs possibles (sensibles à la casse) :
 - `Restic`
 - `Kopia` (par défaut)
 - **Spec.applicationRef :** nom Kubernetes de l'application à sauvegarder.
 - **Spec.appVaultRef:** (*required*) Nom de l'AppVault où le contenu de la sauvegarde doit être stocké.
 - **Spec.backupRetention :** le nombre de sauvegardes à conserver. Zéro indique qu'aucune sauvegarde ne doit être créée.
 - **Spec.snapshotRetention :** le nombre d'instantanés à conserver. Zéro indique qu'aucun snapshot ne doit être créé.
 - **spec.granularity:** la fréquence à laquelle le programme doit s'exécuter. Valeurs possibles, ainsi que les champs associés obligatoires :
 - `hourly` (nécessite que vous spécifiez `spec.minute`)
 - `daily` (nécessite que vous spécifiez `spec.minute` et `spec.hour`)
 - `weekly` (nécessite que vous spécifiez `spec.minute`, `spec.hour`, et `spec.dayOfWeek`)
 - `monthly` (nécessite que vous spécifiez `spec.minute`, `spec.hour`, et `spec.dayOfMonth`)
 - **Spec.dayOfMonth:** (*Optional*) le jour du mois (1 - 31) que le programme doit s'exécuter. Ce champ est obligatoire si la granularité est définie sur `monthly`.
 - **Spec.dayOfWeek:** (*Optional*) le jour de la semaine (0 - 7) que le programme doit s'exécuter. Les valeurs 0 ou 7 indiquent dimanche. Ce champ est obligatoire si la granularité est définie sur `weekly`.
 - **Spec.hour:** (*Optional*) heure du jour (0 - 23) que le programme doit exécuter. Ce champ est obligatoire si la granularité est définie sur `daily`, `weekly` ou `monthly`.
 - **Spec.minute:** (*Optional*) la minute de l'heure (0 - 59) que le programme doit exécuter. Ce champ est obligatoire si la granularité est définie sur `hourly`, `daily`, `weekly` ou `monthly`.


```

---
apiVersion: protect.trident.netapp.io/v1
kind: Schedule
metadata:
  namespace: my-app-namespace
  name: my-cr-name
spec:
  dataMover: Kopia
  applicationRef: my-application
  appVaultRef: appvault-name
  backupRetention: "15"
  snapshotRetention: "15"
  granularity: <monthly>
  dayOfMonth: "1"
  dayOfWeek: "0"
  hour: "0"
  minute: "0"

```

3. Une fois que vous avez rempli le `trident-protect-schedule-cr.yaml` fichier avec les valeurs correctes, appliquez la CR :

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

Créez un planning à l'aide de l'interface de ligne de commandes

Étapes

1. Créez le planning de protection en remplaçant les valeurs entre parenthèses par les informations de votre environnement. Par exemple :



Vous pouvez utiliser `tridentctl-protect create schedule --help` pour afficher les informations d'aide détaillées de cette commande.

```

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>

```

Supprime un snapshot

Supprimez les snapshots programmés ou à la demande dont vous n'avez plus besoin.

Étapes

1. Supprimer l'instantané CR associé à l'instantané :

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

Supprimer une sauvegarde

Supprimez les sauvegardes planifiées ou à la demande qui ne vous sont plus nécessaires.

Étapes

1. Supprimez la CR de sauvegarde associée à la sauvegarde :

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

Vérifier l'état d'une opération de sauvegarde

Vous pouvez utiliser la ligne de commande pour vérifier l'état d'une opération de sauvegarde en cours, terminée ou ayant échoué.

Étapes

1. Utiliser la commande suivante pour récupérer le statut de l'opération de sauvegarde en remplaçant les valeurs entre crochets par des informations de votre environnement :

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

Activez la sauvegarde et la restauration pour les opérations Azure-NetApp-Files (ANF)

Si vous avez installé Trident Protect, vous pouvez activer la fonctionnalité de sauvegarde et de restauration compactes pour les systèmes back-end qui utilisent la classe de stockage Azure-NetApp-Files et qui ont été créés avant Trident 24.06. Cette fonctionnalité fonctionne avec les volumes NFSv4 et ne consomme pas d'espace supplémentaire dans le pool de capacité.

Avant de commencer

Vérifiez les points suivants :

- Vous avez installé Trident Protect.
- Vous avez défini une application dans Trident Protect. Cette application aura une fonctionnalité de protection limitée jusqu'à ce que vous ayez terminé cette procédure.
- Vous avez `azure-netapp-files` sélectionné comme classe de stockage par défaut pour votre système back-end de stockage.

Développez pour les étapes de configuration

1. Si le volume ANF a été créé avant la mise à niveau vers Trident 24.10, procédez comme suit dans Trident :

- a. Activez le répertoire Snapshot pour chaque volume persistant basé sur Azure-NetApp-Files et associé à l'application :

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

- b. Vérifiez que le répertoire de snapshot a été activé pour chaque PV associé :

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

Réponse :

```
snapshotDirectory: "true"
```

+

Lorsque le répertoire de snapshots n'est pas activé, Trident Protect choisit la fonctionnalité de sauvegarde standard, qui consomme temporairement de l'espace dans le pool de capacité pendant le processus de sauvegarde. Dans ce cas, assurez-vous que l'espace disponible dans le pool de capacité est suffisant pour créer un volume temporaire de la taille du volume en cours de sauvegarde.

Résultat

L'application est prête pour la sauvegarde et la restauration à l'aide de Trident Protect. Chaque demande de volume persistant est également disponible pour être utilisée par d'autres applications à des fins de sauvegarde et de restauration.

Restaurez les applications à l'aide de Trident Protect

Vous pouvez utiliser Trident Protect pour restaurer votre application à partir d'une copie Snapshot ou d'une sauvegarde. La restauration d'un snapshot existant est plus rapide lors de la restauration d'une application sur le même cluster.



Lorsque vous restaurez une application, tous les crochets d'exécution configurés pour l'application sont restaurés avec l'application. Si un hook d'exécution post-restauration est présent, il s'exécute automatiquement dans le cadre de l'opération de restauration.

Annotations et étiquettes de namespace pendant les opérations de restauration et de basculement

Lors des opérations de restauration et de basculement, les libellés et les annotations dans l'espace de noms de destination correspondent aux libellés et aux annotations dans l'espace de noms source. Des étiquettes ou des annotations provenant de l'espace de noms source qui n'existent pas dans l'espace de noms de destination sont ajoutées et toutes les étiquettes ou annotations qui existent déjà sont écrasées pour

correspondre à la valeur de l'espace de noms source. Les libellés ou annotations qui existent uniquement dans l'espace de noms de destination restent inchangés.



Si vous utilisez RedHat OpenShift, il est important de noter le rôle critique des annotations d'espace de noms dans les environnements OpenShift. Les annotations de l'espace de noms garantissent que les pods restaurés respectent les autorisations et les configurations de sécurité appropriées définies par les contraintes de contexte de sécurité (CSC) OpenShift et qu'ils peuvent accéder aux volumes sans problèmes d'autorisation. Pour plus d'informations, reportez-vous au ["Documentation sur les contraintes de contexte de sécurité OpenShift"](#).

Vous pouvez empêcher l'écrasement d'annotations spécifiques dans l'espace de noms de destination en configurant la variable d'environnement Kubernetes `RESTORE_SKIP_NAMESPACE_ANNOTATIONS` avant d'effectuer l'opération de restauration ou de basculement. Par exemple :

```
kubectl set env -n trident-protect deploy/trident-protect-controller-manager
RESTORE_SKIP_NAMESPACE_ANNOTATIONS=<annotation_key_to_skip_1>,<annotation_key_to_skip_2>
```

Si vous avez installé l'application source à l'aide de Helm avec l' `--create-namespace` indicateur, un traitement spécial est donné à la `name` clé d'étiquette. Lors du processus de restauration ou de basculement, Trident Protect copie cette étiquette dans l'espace de noms de destination, mais met à jour la valeur vers la valeur de l'espace de noms de destination si la valeur de la source correspond à l'espace de noms source. Si cette valeur ne correspond pas à l'espace de noms source, elle est copiée dans l'espace de noms de destination sans modification.

Exemple

L'exemple suivant présente un espace de noms source et de destination, chacun avec des annotations et des libellés différents. Vous pouvez voir l'état de l'espace de noms de destination avant et après l'opération, ainsi que la manière dont les annotations et les étiquettes sont combinées ou écrasées dans l'espace de noms de destination.

Avant l'opération de restauration ou de basculement

Le tableau suivant illustre l'état de l'exemple d'espaces de noms source et de destination avant l'opération de restauration ou de basculement :

Espace de noms	Annotations	Étiquettes
Espace de noms ns-1 (source)	• annotation.one/key : « updatedvalue » • annotation.deux/touche : « vrai »	• environnement=production • conformité = hipaa • name=ns-1
Espace de noms ns-2 (destination)	• annotation.un/touche : « vrai » • annotation.trois/touche : « false »	• role=base de données

Après l'opération de restauration

Le tableau suivant illustre l'état de l'exemple d'espace de noms de destination après une opération de restauration ou de basculement. Certaines clés ont été ajoutées, d'autres ont été écrasées et le `name` libellé a été mis à jour pour correspondre à l'espace de noms de destination :

Espace de noms	Annotations	Étiquettes
Espace de noms ns-2 (destination)	• <code>annotation.one/key</code> : « <code>updatedvalue</code> » • <code>annotation.deux/touche</code> : « <code>vrai</code> » • <code>annotation.trois/touche</code> : « <code>false</code> »	• <code>name=ns-2</code> • <code>conformité = hipaa</code> • <code>environnement=production</code> • <code>role=base de données</code>

Restauration d'une sauvegarde vers un autre espace de noms

Lorsque vous restaurez une sauvegarde dans un espace de noms différent à l'aide d'une sauvegarde CR BackupRestore, Trident Protect restaure l'application dans un nouvel espace de noms et crée une demande de modification d'application pour l'application restaurée. Pour protéger l'application restaurée, créez des sauvegardes ou des snapshots à la demande ou établissez un calendrier de protection.



La restauration d'une sauvegarde dans un espace de noms différent avec des ressources existantes ne modifie aucune ressource qui partage des noms avec ceux de la sauvegarde. Pour restaurer toutes les ressources de la sauvegarde, supprimez et recréez l'espace de noms cible ou restaurez la sauvegarde dans un nouvel espace de noms.

Utiliser une CR

Étapes

1. Créez le fichier de ressource personnalisée (CR) et nommez-le `trident-protect-backup-restore-cr.yaml`.
2. Dans le fichier que vous avez créé, configurez les attributs suivants :
 - **metadata.name**: (*required*) le nom de cette ressource personnalisée; choisissez un nom unique et sensible pour votre environnement.
 - **Spec.appArchivePath** : chemin d'accès dans AppVault où sont stockés le contenu de la sauvegarde. Vous pouvez utiliser la commande suivante pour trouver ce chemin :

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

- **Spec.appVaultRef**: (*required*) Nom de l'AppVault où sont stockés le contenu de la sauvegarde.
- **spec.namespaceMapping**: mappage de l'espace de noms source de l'opération de restauration sur l'espace de noms de destination. Remplacez `my-source-namespace` et `my-destination-namespace` par des informations provenant de votre environnement.
- **Spec.storageClassMapping** : mappage de la classe de stockage source de l'opération de restauration à la classe de stockage de destination. Remplacez `destinationStorageClass` et `sourceStorageClass` par des informations provenant de votre environnement.

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

3. (*Facultatif*) si vous devez sélectionner uniquement certaines ressources de l'application à restaurer, ajoutez un filtrage qui inclut ou exclut les ressources marquées avec des étiquettes particulières :

- **ResourceFilter.resourceSelectionCriteria**: (Requis pour le filtrage) utiliser `Include` ou `Exclude` inclure ou exclure une ressource définie dans `resourceMatchers`. Ajoutez les paramètres `resourceMatchers` suivants pour définir les ressources à inclure ou à exclure :
 - **ResourceFilter.resourceMatchers** : un tableau d'objets `resourceMatcher`. Si vous définissez plusieurs éléments dans ce tableau, ils correspondent en tant qu'opération OU et les champs

de chaque élément (groupe, type, version) correspondent en tant qu'opération ET.

- **ResourceMatchers[].group:** (*Optional*) Groupe de la ressource à filtrer.
- **ResourceMatchers[].kind:** (*Optional*) Type de la ressource à filtrer.
- **ResourceMatchers[].version:** (*Optional*) version de la ressource à filtrer.
- **ResourceMatchers[].names:** (*Optional*) noms dans le champ Kubernetes metadata.name de la ressource à filtrer.
- **ResourceMatchers[].namespaces:** (*Optional*) Namespaces dans le champ Kubernetes metadata.name de la ressource à filtrer.
- **ResourceMatchers[].labelSelectors:** (*Optional*) chaîne de sélecteur de libellé dans le champ Kubernetes metadata.name de la ressource, comme défini dans le ["Documentation Kubernetes"](#). Par exemple : "trident.netapp.io/os=linux".

Par exemple :

```
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. Une fois que vous avez rempli le trident-protect-backup-restore-cr.yaml fichier avec les valeurs correctes, appliquez la CR :

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

Utilisez l'CLI

Étapes

1. Restaurez la sauvegarde dans un espace de noms différent, en remplaçant les valeurs entre parenthèses par les informations de votre environnement. L' namespace-mapping`argument utilise des espaces de noms séparés par deux-points pour mapper les espaces de noms source aux espaces de noms de destination corrects dans le format `source1:dest1,source2:dest2. Par exemple :

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

Restaurer à partir d'une sauvegarde vers l'espace de noms d'origine

Vous pouvez à tout moment restaurer une sauvegarde dans l'espace de noms d'origine.

Utiliser une CR

Étapes

1. Créez le fichier de ressource personnalisée (CR) et nommez-le `trident-protect-backup-ipr-cr.yaml`.
2. Dans le fichier que vous avez créé, configurez les attributs suivants :

- **metadata.name:** (*required*) le nom de cette ressource personnalisée; choisissez un nom unique et sensible pour votre environnement.
- **Spec.appArchivePath** : chemin d'accès dans AppVault où sont stockés le contenu de la sauvegarde. Vous pouvez utiliser la commande suivante pour trouver ce chemin :

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

- **Spec.appVaultRef:** (*required*) Nom de l'AppVault où sont stockés le contenu de la sauvegarde.

Par exemple :

```
---
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. (*Facultatif*) si vous devez sélectionner uniquement certaines ressources de l'application à restaurer, ajoutez un filtrage qui inclut ou exclut les ressources marquées avec des étiquettes particulières :

- **ResourceFilter.resourceSelectionCriteria:** (Requis pour le filtrage) utiliser `Include` ou `Exclude` inclure ou exclure une ressource définie dans `resourceMatchers`. Ajoutez les paramètres `resourceMatchers` suivants pour définir les ressources à inclure ou à exclure :
 - **ResourceFilter.resourceMatchers** : un tableau d'objets `resourceMatcher`. Si vous définissez plusieurs éléments dans ce tableau, ils correspondent en tant qu'opération OU et les champs de chaque élément (groupe, type, version) correspondent en tant qu'opération ET.
 - **ResourceMatchers[].group:** (*Optional*) Groupe de la ressource à filtrer.
 - **ResourceMatchers[].kind:** (*Optional*) Type de la ressource à filtrer.
 - **ResourceMatchers[].version:** (*Optional*) version de la ressource à filtrer.
 - **ResourceMatchers[].names:** (*Optional*) noms dans le champ Kubernetes `metadata.name` de la ressource à filtrer.
 - **ResourceMatchers[].namespaces:** (*Optional*) Namespaces dans le champ Kubernetes `metadata.name` de la ressource à filtrer.

- **ResourceMatchers[].labelSelectors:** (*Optional*) chaîne de sélecteur de libellé dans le champ Kubernetes metadata.name de la ressource, comme défini dans le ["Documentation Kubernetes"](#). Par exemple : "trident.netapp.io/os=linux".

Par exemple :

```
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. Une fois que vous avez rempli le trident-protect-backup-ipr-cr.yaml fichier avec les valeurs correctes, appliquez la CR :

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

Utilisez l'CLI

Étapes

1. Restaurez la sauvegarde dans l'espace de noms d'origine en remplaçant les valeurs entre parenthèses par les informations de votre environnement. L'backup`argument utilise un nom d'espace de noms et un nom de sauvegarde au format `

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

Restauration à partir d'une sauvegarde sur un autre cluster

Vous pouvez restaurer une sauvegarde sur un autre cluster en cas de problème avec le cluster d'origine.

Avant de commencer

Assurez-vous que les conditions préalables suivantes sont remplies :

- Trident Protect est installé sur le cluster de destination.
- Le cluster de destination a accès au chemin de compartiment du même AppVault que le cluster source, où la sauvegarde est stockée.

Étapes

1. Vérifier la disponibilité du CR AppVault sur le cluster de destination à l'aide du plug-in CLI Trident Protect :

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



Assurez-vous que l'espace de noms destiné à la restauration d'application existe sur le cluster de destination.

2. Afficher le contenu de la sauvegarde de l'AppVault disponible à partir du cluster de destination :

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

L'exécution de cette commande affiche les sauvegardes disponibles dans le AppVault, y compris leurs clusters d'origine, les noms d'applications correspondants, les horodatages et les chemins d'archivage.

Exemple de sortie :

```
+-----+-----+-----+-----+
+-----+-----+
|  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. Restaurez l'application sur le cluster de destination à l'aide du nom AppVault et du chemin d'archivage :

Utiliser une CR

1. Créez le fichier de ressource personnalisée (CR) et nommez-le `trident-protect-backup-restore-cr.yaml`.
2. Dans le fichier que vous avez créé, configurez les attributs suivants :
 - **metadata.name**: (*required*) le nom de cette ressource personnalisée; choisissez un nom unique et sensible pour votre environnement.
 - **Spec.appVaultRef**: (*required*) Nom de l'AppVault où sont stockés le contenu de la sauvegarde.
 - **Spec.appArchivePath** : chemin d'accès dans AppVault où sont stockés le contenu de la sauvegarde. Vous pouvez utiliser la commande suivante pour trouver ce chemin :

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



Si BackupRestore CR n'est pas disponible, vous pouvez utiliser la commande mentionnée à l'étape 2 pour afficher le contenu de la sauvegarde.

- **spec.namespaceMapping**: mappage de l'espace de noms source de l'opération de restauration sur l'espace de noms de destination. Remplacez `my-source-namespace` et `my-destination-namespace` par des informations provenant de votre environnement.

Par exemple :

```
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. Une fois que vous avez rempli le `trident-protect-backup-restore-cr.yaml` fichier avec les valeurs correctes, appliquez la CR :

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

Utilisez l'CLI

1. Utilisez la commande suivante pour restaurer l'application, en remplaçant les valeurs entre parenthèses par les informations de votre environnement. L'argument `namespace-mapping` utilise des espaces de noms séparés par deux points pour mapper les espaces de noms source aux

espaces de noms de destination corrects au format source1:dest1,source2:dest2. Par exemple :

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

Restauration d'un snapshot vers un autre espace de noms

Vous pouvez restaurer les données d'un instantané à l'aide d'un fichier de ressource personnalisée (CR) dans un espace de noms différent ou dans l'espace de noms source d'origine. Lorsque vous restaurez un snapshot dans un espace de noms différent à l'aide d'un CR SnapshotRestore, Trident Protect restaure l'application dans un nouvel espace de noms et crée une CR d'application pour l'application restaurée. Pour protéger l'application restaurée, créez des sauvegardes ou des snapshots à la demande ou établissez un calendrier de protection.

Utiliser une CR

Étapes

1. Créez le fichier de ressource personnalisée (CR) et nommez-le `trident-protect-snapshot-restore-cr.yaml`.
2. Dans le fichier que vous avez créé, configurez les attributs suivants :
 - **metadata.name:** (*required*) le nom de cette ressource personnalisée; choisissez un nom unique et sensible pour votre environnement.
 - **Spec.appVaultRef:** (*required*) le nom du AppVault dans lequel le contenu de l'instantané est stocké.
 - **Spec.appArchivePath :** chemin d'accès dans AppVault où sont stockés le contenu de l'instantané. Vous pouvez utiliser la commande suivante pour trouver ce chemin :

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

- **spec.namespaceMapping:** mappage de l'espace de noms source de l'opération de restauration sur l'espace de noms de destination. Remplacez `my-source-namespace` et `my-destination-namespace` par des informations provenant de votre environnement.
- **Spec.storageClassMapping :** mappage de la classe de stockage source de l'opération de restauration à la classe de stockage de destination. Remplacez `destinationStorageClass` et `sourceStorageClass` par des informations provenant de votre environnement.

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

3. (*Facultatif*) si vous devez sélectionner uniquement certaines ressources de l'application à restaurer, ajoutez un filtrage qui inclut ou exclut les ressources marquées avec des étiquettes particulières :
 - **ResourceFilter.resourceSelectionCriteria:** (Requis pour le filtrage) utiliser `Include` ou `Exclude` inclure ou exclure une ressource définie dans `resourceMatchers`. Ajoutez les paramètres `resourceMatchers` suivants pour définir les ressources à inclure ou à exclure :

- **ResourceFilter.resourceMatchers** : un tableau d'objets resourceMatcher. Si vous définissez plusieurs éléments dans ce tableau, ils correspondent en tant qu'opération OU et les champs de chaque élément (groupe, type, version) correspondent en tant qu'opération ET.
 - **ResourceMatchers[].group**: (*Optional*) Groupe de la ressource à filtrer.
 - **ResourceMatchers[].kind**: (*Optional*) Type de la ressource à filtrer.
 - **ResourceMatchers[].version**: (*Optional*) version de la ressource à filtrer.
 - **ResourceMatchers[].names**: (*Optional*) noms dans le champ Kubernetes metadata.name de la ressource à filtrer.
 - **ResourceMatchers[].namespaces**: (*Optional*) Namespaces dans le champ Kubernetes metadata.name de la ressource à filtrer.
 - **ResourceMatchers[].labelSelectors**: (*Optional*) chaîne de sélecteur de libellé dans le champ Kubernetes metadata.name de la ressource, comme défini dans le ["Documentation Kubernetes"](#). Par exemple : "trident.netapp.io/os=linux".

Par exemple :

```
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. Une fois que vous avez rempli le trident-protect-snapshot-restore-cr.yaml fichier avec les valeurs correctes, appliquez la CR :

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

Utilisez l'CLI

Étapes

1. Restaurez l'instantané dans un autre espace de noms, en remplaçant les valeurs entre parenthèses par les informations de votre environnement.
 - L'argument snapshot utilise un nom d'espace de noms et un nom

d'instantané au format ``<namespace>/<name>`.

- L'argument `namespace-mapping` utilise des espaces de noms séparés par deux-points pour mapper les espaces de noms source aux espaces de noms de destination corrects dans le format `source1:dest1,source2:dest2`.

Par exemple :

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

Restaurer à partir d'un snapshot vers l'espace de noms d'origine

Vous pouvez à tout moment restaurer un snapshot dans l'espace de noms d'origine.

Utiliser une CR

Étapes

1. Créez le fichier de ressource personnalisée (CR) et nommez-le `trident-protect-snapshot-ipr-cr.yaml`.
2. Dans le fichier que vous avez créé, configurez les attributs suivants :
 - **metadata.name:** (*required*) le nom de cette ressource personnalisée; choisissez un nom unique et sensible pour votre environnement.
 - **Spec.appVaultRef:** (*required*) le nom du AppVault dans lequel le contenu de l'instantané est stocké.
 - **Spec.appArchivePath :** chemin d'accès dans AppVault où sont stockés le contenu de l'instantané. Vous pouvez utiliser la commande suivante pour trouver ce chemin :

```
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. (*Facultatif*) si vous devez sélectionner uniquement certaines ressources de l'application à restaurer, ajoutez un filtrage qui inclut ou exclut les ressources marquées avec des étiquettes particulières :
 - **ResourceFilter.resourceSelectionCriteria:** (Requis pour le filtrage) utiliser `Include` ou `Exclude` inclure ou exclure une ressource définie dans `resourceMatchers`. Ajoutez les paramètres `resourceMatchers` suivants pour définir les ressources à inclure ou à exclure :
 - **ResourceFilter.resourceMatchers :** un tableau d'objets `resourceMatcher`. Si vous définissez plusieurs éléments dans ce tableau, ils correspondent en tant qu'opération OU et les champs de chaque élément (groupe, type, version) correspondent en tant qu'opération ET.
 - **ResourceMatchers[].group:** (*Optional*) Groupe de la ressource à filtrer.
 - **ResourceMatchers[].kind:** (*Optional*) Type de la ressource à filtrer.
 - **ResourceMatchers[].version:** (*Optional*) version de la ressource à filtrer.
 - **ResourceMatchers[].names:** (*Optional*) noms dans le champ Kubernetes `metadata.name` de la ressource à filtrer.
 - **ResourceMatchers[].namespaces:** (*Optional*) Namespaces dans le champ Kubernetes `metadata.name` de la ressource à filtrer.
 - **ResourceMatchers[].labelSelectors:** (*Optional*) chaîne de sélecteur de libellé dans le

champ Kubernetes metadata.name de la ressource, comme défini dans le ["Documentation Kubernetes"](#). Par exemple : "trident.netapp.io/os=linux".

Par exemple :

```
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. Une fois que vous avez rempli le trident-protect-snapshot-ipr-cr.yaml fichier avec les valeurs correctes, appliquez la CR :

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

Utilisez l'CLI

Étapes

1. Restaurez l'instantané dans l'espace de noms d'origine en remplaçant les valeurs entre parenthèses par les informations de votre environnement. Par exemple :

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

Vérifiez l'état d'une opération de restauration

Vous pouvez utiliser la ligne de commande pour vérifier l'état d'une opération de restauration en cours, terminée ou ayant échoué.

Étapes

1. Utilisez la commande suivante pour récupérer le statut de l'opération de restauration en remplaçant les valeurs entre crochets par des informations de votre environnement :

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

Répliquez des applications à l'aide de NetApp SnapMirror et Trident Protect

Avec Trident Protect, vous pouvez utiliser les fonctionnalités de réplication asynchrone de la technologie NetApp SnapMirror pour répliquer les modifications des données et des applications d'un système back-end à un autre, sur le même cluster ou entre différents clusters.

Annotations et étiquettes de namespace pendant les opérations de restauration et de basculement

Lors des opérations de restauration et de basculement, les libellés et les annotations dans l'espace de noms de destination correspondent aux libellés et aux annotations dans l'espace de noms source. Des étiquettes ou des annotations provenant de l'espace de noms source qui n'existent pas dans l'espace de noms de destination sont ajoutées et toutes les étiquettes ou annotations qui existent déjà sont écrasées pour correspondre à la valeur de l'espace de noms source. Les libellés ou annotations qui existent uniquement dans l'espace de noms de destination restent inchangés.



Si vous utilisez RedHat OpenShift, il est important de noter le rôle critique des annotations d'espace de noms dans les environnements OpenShift. Les annotations de l'espace de noms garantissent que les pods restaurés respectent les autorisations et les configurations de sécurité appropriées définies par les contraintes de contexte de sécurité (CSC) OpenShift et qu'ils peuvent accéder aux volumes sans problèmes d'autorisation. Pour plus d'informations, reportez-vous au ["Documentation sur les contraintes de contexte de sécurité OpenShift"](#).

Vous pouvez empêcher l'écrasement d'annotations spécifiques dans l'espace de noms de destination en configurant la variable d'environnement Kubernetes `RESTORE_SKIP_NAMESPACE_ANNOTATIONS` avant d'effectuer l'opération de restauration ou de basculement. Par exemple :

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

Si vous avez installé l'application source à l'aide de Helm avec l'indicateur `--create-namespace`, un traitement spécial est donné à la clé d'étiquette `name`. Lors du processus de restauration ou de basculement, Trident Protect copie cette étiquette dans l'espace de noms de destination, mais met à jour la valeur vers la valeur de l'espace de noms de destination si la valeur de la source correspond à l'espace de noms source. Si cette valeur ne correspond pas à l'espace de noms source, elle est copiée dans l'espace de noms de destination sans modification.

Exemple

L'exemple suivant présente un espace de noms source et de destination, chacun avec des annotations et des libellés différents. Vous pouvez voir l'état de l'espace de noms de destination avant et après l'opération, ainsi que la manière dont les annotations et les étiquettes sont combinées ou écrasées dans l'espace de noms de destination.

Avant l'opération de restauration ou de basculement

Le tableau suivant illustre l'état de l'exemple d'espaces de noms source et de destination avant l'opération de restauration ou de basculement :

Espace de noms	Annotations	Étiquettes
Espace de noms ns-1 (source)	• annotation.one/key : « updatedvalue » • annotation.deux/touche : « vrai »	• environnement=production • conformité = hipaa • name=ns-1
Espace de noms ns-2 (destination)	• annotation.un/touche : « vrai » • annotation.trois/touche : « false »	• role=base de données

Après l'opération de restauration

Le tableau suivant illustre l'état de l'exemple d'espace de noms de destination après une opération de restauration ou de basculement. Certaines clés ont été ajoutées, d'autres ont été écrasées et le `name` libellé a été mis à jour pour correspondre à l'espace de noms de destination :

Espace de noms	Annotations	Étiquettes
Espace de noms ns-2 (destination)	• annotation.one/key : « updatedvalue » • annotation.deux/touche : « vrai » • annotation.trois/touche : « false »	• name=ns-2 • conformité = hipaa • environnement=production • role=base de données



Vous pouvez configurer Trident Protect pour qu'il bloque et dégèle les systèmes de fichiers pendant les opérations de protection des données. ["En savoir plus sur la configuration de la congélation du système de fichiers avec Trident Protect"](#).

Configuration d'une relation de réplication

La configuration d'une relation de réplication implique les éléments suivants :

- Choix de la fréquence à laquelle Trident Protect doit créer une copie Snapshot d'application (qui inclut les ressources Kubernetes de l'application ainsi que les snapshots de volume pour chacun des volumes de l'application)
- Choix de la planification de la réplication (inclut les ressources Kubernetes ainsi que les données de volume persistant)
- Définition de la durée de prise de l'instantané

Étapes

1. Créez un AppVault pour l'application source sur le cluster source. Selon votre fournisseur de stockage, modifiez un exemple en fonction de ["Ressources personnalisées AppVault"](#) votre environnement :

Créez un AppVault à l'aide d'une CR

- a. Créez le fichier de ressource personnalisée (CR) et nommez-le (par exemple, `trident-protect-appvault-primary-source.yaml`).
- b. Configurez les attributs suivants :
 - **metadata.name:** (*required*) le nom de la ressource personnalisée AppVault. Notez le nom que vous choisissez, car les autres fichiers CR nécessaires pour une relation de réplication font référence à cette valeur.
 - **spec.providerConfig:** (*required*) stocke la configuration nécessaire pour accéder à AppVault à l'aide du fournisseur spécifié. Choisissez un nom de bucketName et tout autre détail nécessaire pour votre fournisseur. Notez les valeurs que vous choisissez, car les autres fichiers CR nécessaires à une relation de réplication font référence à ces valeurs. Reportez-vous à la section "[Ressources personnalisées AppVault](#)" pour obtenir des exemples de CRS AppVault avec d'autres fournisseurs.
 - **spec.providerCredentials:** (*required*) stocke les références à toute information d'identification requise pour accéder à AppVault à l'aide du fournisseur spécifié.
 - **spec.providerCredentials.valueFromSecret:** (*required*) indique que la valeur d'identification doit provenir d'un secret.
 - **Key:** (*required*) la clé valide du secret à sélectionner.
 - **Name:** (*required*) Nom du secret contenant la valeur de ce champ. Doit être dans le même espace de noms.
 - **spec.providerCredentials.secretAccessKey:** (*required*) la clé d'accès utilisée pour accéder au fournisseur. Le **nom** doit correspondre à **spec.providerCredentials.valueFromSecret.name**.
 - **spec.providerType:** (*required*) détermine ce qui permet la sauvegarde, par exemple NetApp ONTAP S3, S3 générique, Google Cloud ou Microsoft Azure. Valeurs possibles :
 - aws
 - azure
 - gcp
 - générique-s3
 - ONTAP s3
 - StorageGRID s3
- c. Une fois que vous avez rempli le `trident-protect-appvault-primary-source.yaml` fichier avec les valeurs correctes, appliquez la CR :

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

Créez un AppVault à l'aide de la CLI

- a. Créez AppVault, en remplaçant les valeurs entre parenthèses par les informations de votre environnement :

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

2. Créez l'application source CR :

Créez l'application source à l'aide d'une demande de modification

- a. Créez le fichier de ressource personnalisée (CR) et nommez-le (par exemple, `trident-protect-app-source.yaml`).
- b. Configurez les attributs suivants :
 - **metadata.name:** (*required*) le nom de la ressource personnalisée de l'application. Notez le nom que vous choisissez, car les autres fichiers CR nécessaires pour une relation de réplication font référence à cette valeur.
 - **spec.includedNamespaces:** (*required*) un tableau d'espaces de noms et d'étiquettes associées. Utilisez des noms d'espace de noms et, éventuellement, affinez la portée des espaces de noms avec des étiquettes pour spécifier les ressources qui existent dans les espaces de noms répertoriés ici. L'espace de nom de l'application doit faire partie de ce tableau.

Exemple 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. Une fois que vous avez rempli le `trident-protect-app-source.yaml` fichier avec les valeurs correctes, appliquez la CR :

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

Créez l'application source à l'aide de l'interface de ligne de commande

- a. Créez l'application source. Par exemple :

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

3. Vous pouvez également créer un snapshot de l'application source. Ce snapshot est utilisé comme base pour l'application sur le cluster de destination. Si vous ignorez cette étape, vous devez attendre l'exécution du prochain snapshot planifié pour avoir un instantané récent.

Prendre un instantané à l'aide d'une CR

a. Créez un planning de réplication pour l'application source :

- i. Créez le fichier de ressource personnalisée (CR) et nommez-le (par exemple, `trident-protect-schedule.yaml`).
- ii. Configurez les attributs suivants :
 - **metadata.name**: (*required*) le nom de la ressource personnalisée d'horaire.
 - **Spec.AppVaultRef**: (*required*) cette valeur doit correspondre au champ `metadata.name` de l'AppVault pour l'application source.
 - **Spec.ApplicationRef**: (*required*) cette valeur doit correspondre au champ `metadata.name` de l'application source CR.
 - **Spec.backupRetention**: (*required*) ce champ est obligatoire et la valeur doit être définie sur 0.
 - **Spec.enabled** : doit être défini sur `true`.
 - **spec.granularity**: doit être défini sur `Custom`.
 - **Spec.recurrenceRule** : définissez une date de début en heure UTC et un intervalle de récurrence.
 - **Spec.snapshotRetention** : doit être défini sur 2.

Exemple YAML :

```
---
apiVersion: protect.trident.netapp.io/v1
kind: Schedule
metadata:
  name: appmirror-schedule-0e1f88ab-f013-4bce-8ae9-6afed9df59a1
  namespace: my-app-namespace
spec:
  appVaultRef: generic-s3-trident-protect-src-bucket-04b6b4ec-
46a3-420a-b351-45795e1b5e34
  applicationRef: my-app-name
  backupRetention: "0"
  enabled: true
  granularity: custom
  recurrenceRule: |-
    DTSTART:20220101T000200Z
    RRULE:FREQ=MINUTELY;INTERVAL=5
  snapshotRetention: "2"
```

- i. Une fois que vous avez rempli le `trident-protect-schedule.yaml` fichier avec les valeurs correctes, appliquez la CR :


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

Créer un snapshot à l'aide de l'interface de ligne de commande

- a. Créez l'instantané, en remplaçant les valeurs entre parenthèses par les informations de votre environnement. Par exemple :

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

4. Créez une application source AppVault CR sur le cluster de destination qui est identique à la CR AppVault que vous avez appliquée sur le cluster source et nommez-la (par exemple, trident-protect-appvault-primary-destination.yaml).

5. Appliquer la CR :

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

6. Créez un AppVault pour l'application de destination sur le cluster de destination. Selon votre fournisseur de stockage, modifiez un exemple en fonction de "[Ressources personnalisées AppVault](#)" votre environnement :

- a. Créez le fichier de ressource personnalisée (CR) et nommez-le (par exemple, trident-protect-appvault-secondary-destination.yaml).
- b. Configurez les attributs suivants :
 - **metadata.name:** (*required*) le nom de la ressource personnalisée AppVault. Notez le nom que vous choisissez, car les autres fichiers CR nécessaires pour une relation de réplication font référence à cette valeur.
 - **spec.providerConfig:** (*required*) stocke la configuration nécessaire pour accéder à AppVault à l'aide du fournisseur spécifié. Choisissez un `bucketName` et d'autres détails nécessaires pour votre fournisseur. Notez les valeurs que vous choisissez, car les autres fichiers CR nécessaires à une relation de réplication font référence à ces valeurs. Reportez-vous à la section "[Ressources personnalisées AppVault](#)" pour obtenir des exemples de CRS AppVault avec d'autres fournisseurs.
 - **spec.providerCredentials:** (*required*) stocke les références à toute information d'identification requise pour accéder à AppVault à l'aide du fournisseur spécifié.
 - **spec.providerCredentials.valueFromSecret:** (*required*) indique que la valeur d'identification doit provenir d'un secret.
 - **Key:** (*required*) la clé valide du secret à sélectionner.
 - **Name:** (*required*) Nom du secret contenant la valeur de ce champ. Doit être dans le même espace de noms.
 - **spec.providerCredentials.secretAccessKey:** (*required*) la clé d'accès utilisée pour accéder

au fournisseur. Le **nom** doit correspondre à **spec.providerCredentials.valueFromSecret.name**.

- **spec.providerType:** (*required*) détermine ce qui permet la sauvegarde, par exemple NetApp ONTAP S3, S3 générique, Google Cloud ou Microsoft Azure. Valeurs possibles :

- aws
- azure
- gcp
- générique-s3
- ONTAP s3
- StorageGRID s3

- c. Une fois que vous avez rempli le `trident-protect-appvault-secondary-destination.yaml` fichier avec les valeurs correctes, appliquez la CR :

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

7. Créez un fichier CR AppMirrorRelationship :

Créez un AppMirrorRelationship à l'aide d'une CR

- a. Créez le fichier de ressource personnalisée (CR) et nommez-le (par exemple, trident-protect-relationship.yaml).
- b. Configurez les attributs suivants :
 - **metadata.name:** (obligatoire) le nom de la ressource personnalisée AppMirrorRelationship.
 - **spec.destinationAppVaultRef:** (*required*) cette valeur doit correspondre au nom de l'AppVault pour l'application de destination sur le cluster de destination.
 - **spec.namespaceMapping:** (*required*) les espaces de noms de destination et de source doivent correspondre à l'espace de noms d'application défini dans la CR de l'application correspondante.
 - **Spec.sourceAppVaultRef:** (*required*) cette valeur doit correspondre au nom du AppVault pour l'application source.
 - **Spec.sourceApplicationName:** (*required*) cette valeur doit correspondre au nom de l'application source que vous avez définie dans la CR de l'application source.
 - **Spec.storageClassName:** (*required*) Choisissez le nom d'une classe de stockage valide sur le cluster. La classe de stockage doit être liée à une VM de stockage ONTAP utilisée par peering avec l'environnement source.
 - **Spec.recurrenceRule :** définissez une date de début en heure UTC et un intervalle de récurrence.

Exemple 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-vsims-2
```

- c. Une fois que vous avez rempli le `trident-protect-relationship.yaml` fichier avec les valeurs correctes, appliquez la CR :

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

Créez un AppMirrorRelationship à l'aide de l'interface de ligne de commande

- a. Créez et appliquez l'objet AppMirrorRelationship, en remplaçant les valeurs entre parenthèses par les informations de votre environnement. Par exemple :

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

8. (Optional) Vérifiez l'état et l'état de la relation de réplication :

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

Basculement vers le cluster de destination

À l'aide de Trident Protect, vous pouvez basculer les applications répliquées vers un cluster de destination. Cette procédure arrête la relation de réplication et met l'application en ligne sur le cluster de destination. Trident Protect n'arrête pas l'application sur le cluster source si celle-ci était opérationnelle.

Étapes

1. Ouvrez le fichier CR AppMirrorRelationship (par exemple, `trident-protect-relationship.yaml`) et définissez la valeur de **spec.desiredState** sur `Promoted`.
2. Enregistrez le fichier CR.
3. Appliquez la CR :

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

4. (Facultatif) Créez les plannings de protection dont vous avez besoin sur l'application ayant fait l'objet d'un basculement.
5. (Optional) Vérifiez l'état et l'état de la relation de réplication :

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

Resynchronisation d'une relation de réplication ayant échoué

L'opération de resynchronisation rétablit la relation de réplication. Une fois l'opération de resynchronisation effectuée, l'application source d'origine devient l'application en cours d'exécution et toutes les modifications apportées à l'application en cours d'exécution sur le cluster de destination sont supprimées.

Le processus arrête l'application sur le cluster de destination avant de rétablir la réplication.



Toutes les données écrites sur l'application de destination pendant le basculement sont perdues.

Étapes

1. Créez un instantané de l'application source.
2. Ouvrez le fichier CR AppMirrorRelationship (par exemple, `trident-protect-relationship.yaml`) et définissez la valeur `spec.desiredState` sur `Established`.
3. Enregistrez le fichier CR.
4. Appliquez la CR :

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

5. Si vous avez créé des plannings de protection sur le cluster de destination pour protéger l'application en panne, supprimez-les. Toute planification qui reste à l'origine de défaillances des snapshots de volume.

Inversion de la resynchronisation d'une relation de réplication ayant échoué

Lorsque vous inversez la resynchronisation d'une relation de réplication ayant fait l'objet d'un basculement, l'application de destination devient l'application source et la source devient la destination. Les modifications apportées à l'application de destination pendant le basculement sont conservées.

Étapes

1. Supprimez la CR AppMirrorRelationship sur le cluster de destination d'origine. La destination devient alors la source. S'il reste des plannings de protection sur le nouveau cluster de destination, supprimez-les.
2. Configurez une relation de réplication en appliquant les fichiers CR que vous avez utilisés à l'origine pour configurer la relation aux clusters opposés.
3. Assurez-vous que les CRS AppVault sont prêts sur chaque cluster.
4. Configurez une relation de réplication sur le cluster opposé, en configurant les valeurs pour la direction inverse.

Inverser le sens de réplication de l'application

Lorsque vous inversez le sens de la réplication, Trident Protect déplace l'application vers le back-end de stockage de destination tout en continuant à répliquer à nouveau vers le back-end de stockage source d'origine. Trident Protect arrête l'application source et réplique les données vers la destination avant de

basculer vers l'application cible.

Dans ce cas, vous permutez la source et la destination.

Étapes

1. Créer un instantané d'arrêt :

Créez un instantané d'arrêt à l'aide d'une CR

- a. Désactivez les plannings de stratégie de protection pour l'application source.
- b. Créer un fichier ShutdownSnapshot CR :
 - i. Créez le fichier de ressource personnalisée (CR) et nommez-le (par exemple, trident-protect-shutdownsnapshot.yaml).
 - ii. Configurez les attributs suivants :
 - **metadata.name:** (*required*) le nom de la ressource personnalisée.
 - **Spec.AppVaultRef:** (*required*) cette valeur doit correspondre au champ metadata.name de l'AppVault pour l'application source.
 - **Spec.ApplicationRef:** (*required*) cette valeur doit correspondre au champ metadata.name du fichier CR de l'application source.

Exemple 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. Une fois que vous avez rempli le trident-protect-shutdownsnapshot.yaml fichier avec les valeurs correctes, appliquez la CR :

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

Créer un snapshot d'arrêt à l'aide de l'interface de ligne de commandes

- a. Créez l'instantané d'arrêt, en remplaçant les valeurs entre parenthèses par les informations de votre environnement. Par exemple :

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

2. Une fois le snapshot terminé, obtenez l'état du snapshot :

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

3. Recherchez la valeur de **shutdownsnapshot.status.appArchivePath** à l'aide de la commande suivante et enregistrez la dernière partie du chemin d'accès au fichier (également appelée nom de base ; ce sera tout après la dernière barre oblique) :

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

4. Effectuez un basculement du cluster de destination vers le cluster source, avec la modification suivante :



À l'étape 2 de la procédure de basculement, incluez le `spec.promotedSnapshot` champ dans le fichier CR AppMirrorRelationship et définissez sa valeur sur le nom de base que vous avez enregistré à l'étape 3 ci-dessus.

5. Effectuez les étapes de resynchronisation inverse dans [Inversion de la resynchronisation d'une relation de réplication ayant échoué](#).
6. Activez les plannings de protection sur le nouveau cluster source.

Résultat

Les actions suivantes se produisent en raison de la réplication inverse :

- Une copie Snapshot des ressources Kubernetes de l'application source d'origine est effectuée.
- Les pods de l'application source d'origine sont « interrompus » en supprimant les ressources Kubernetes de l'application (laissant les demandes de volume persistant et les volumes persistants en place).
- Une fois les pods arrêtés, des copies Snapshot des volumes de l'application sont prises et répliquées.
- Les relations SnapMirror sont rompues, les volumes de destination étant prêts pour la lecture/l'écriture.
- Les ressources Kubernetes de l'application sont restaurées à partir du snapshot de pré-arrêt, à l'aide des données du volume répliquées après la fermeture de l'application source d'origine.
- La réplication est rétablie dans la direction inverse.

Rétablir le fonctionnement des applications sur le cluster source d'origine

Grâce à Trident Protect, vous pouvez obtenir le « retour arrière » après un basculement en suivant la séquence suivante. Dans ce flux de travail pour restaurer le sens de réplication d'origine, Trident Protect réplique (resyncs) toute modification d'application vers l'application source d'origine avant d'inverser le sens de réplication.

Ce processus commence à partir d'une relation qui a effectué un basculement vers une destination et implique les étapes suivantes :

- Commencer par un état de basculement défaillant.
- Resynchronisez la relation de réplication en sens inverse.



N'effectuez pas d'opération de resynchronisation normale, car cela vous permettra d'ignorer les données écrites sur le cluster de destination pendant la procédure de basculement.

- Inversez le sens de réplication.

Étapes

1. Effectuer les [Inversion de la resynchronisation d'une relation de réplication ayant échoué](#) étapes.
2. Effectuer les [Inverser le sens de réplication de l'application](#) étapes.

Supprimer une relation de réplication

Vous pouvez supprimer une relation de réplication à tout moment. Lorsque vous supprimez la relation de réplication d'application, deux applications distinctes n'ont aucune relation entre elles.

Étapes

1. Supprimez la CR d'AppMirrorRelationship :

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

Migrez vos applications à l'aide de Trident Protect

Vous pouvez migrer vos applications entre des clusters ou des classes de stockage en restaurant vos données de sauvegarde ou d'instantané sur un autre cluster ou une autre classe de stockage.



Lorsque vous migrez une application, tous les crochets d'exécution configurés pour l'application sont migrés avec l'application. Si un hook d'exécution post-restauration est présent, il s'exécute automatiquement dans le cadre de l'opération de restauration.

Opérations de sauvegarde et de restauration

Pour effectuer des opérations de sauvegarde et de restauration dans les scénarios suivants, vous pouvez automatiser des tâches de sauvegarde et de restauration spécifiques.

Clone dans le même cluster

Pour cloner une application sur le même cluster, créez un Snapshot ou sauvegardez et restaurez les données sur le même cluster.

Étapes

1. Effectuez l'une des opérations suivantes :
 - a. ["Créer un snapshot"](#).
 - b. ["Créer une sauvegarde"](#).
2. Sur le même cluster, effectuez l'une des opérations suivantes, selon que vous avez créé un snapshot ou une sauvegarde :
 - a. ["Restaurez vos données à partir du snapshot"](#).
 - b. ["Restaurez vos données à partir de la sauvegarde"](#).

Cloner vers un autre cluster

Pour cloner une application sur un autre cluster (effectuez un clone entre clusters), créez une sauvegarde sur le cluster source, puis restaurez la sauvegarde sur un autre cluster. Assurez-vous que Trident Protect est installé sur le cluster de destination.



Vous pouvez répliquer une application entre différents clusters à l'aide de ["Réplication SnapMirror"](#).

Étapes

1. ["Créer une sauvegarde"](#).
2. Assurez-vous que la CR AppVault du compartiment de stockage objet contenant la sauvegarde a été configurée sur le cluster de destination.
3. Sur le cluster de destination, ["restaurez vos données à partir de la sauvegarde"](#).

Migration des applications d'une classe de stockage vers une autre

Vous pouvez migrer des applications d'une classe de stockage vers une autre classe de stockage en restaurant un snapshot sur la classe de stockage de destination différente.

Par exemple (à l'exclusion des secrets de la CR de restauration) :

```
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
```

Restaurez l'instantané à l'aide d'une CR

Étapes

1. Créez le fichier de ressource personnalisée (CR) et nommez-le `trident-protect-snapshot-restore-cr.yaml`.
2. Dans le fichier que vous avez créé, configurez les attributs suivants :
 - **metadata.name:** (*required*) le nom de cette ressource personnalisée; choisissez un nom unique et sensible pour votre environnement.
 - **Spec.appArchivePath** : chemin d'accès dans AppVault où sont stockés le contenu de l'instantané. Vous pouvez utiliser la commande suivante pour trouver ce chemin :

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

- **Spec.appVaultRef:** (*required*) le nom du AppVault dans lequel le contenu de l'instantané est stocké.
- **spec.namespaceMapping:** mappage de l'espace de noms source de l'opération de restauration sur l'espace de noms de destination. Remplacez `my-source-namespace` et `my-destination-namespace` par des informations provenant de votre environnement.

```
---
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. Si vous avez besoin de sélectionner uniquement certaines ressources de l'application à restaurer, ajoutez un filtrage qui inclut ou exclut les ressources marquées d'étiquettes particulières :
 - **ResourceFilter.resourceSelectionCriteria:** (Requis pour le filtrage) utilisez `include` or `exclude` pour inclure ou exclure une ressource définie dans `resourceMatchers`. Ajoutez les paramètres `resourceMatchers` suivants pour définir les ressources à inclure ou à exclure :
 - **ResourceFilter.resourceMatchers** : un tableau d'objets `resourceMatcher`. Si vous définissez plusieurs éléments dans ce tableau, ils correspondent en tant qu'opération OU et les champs de chaque élément (groupe, type, version) correspondent en tant qu'opération ET.
 - **ResourceMatchers[].group:** (*Optional*) Groupe de la ressource à filtrer.
 - **ResourceMatchers[].kind:** (*Optional*) Type de la ressource à filtrer.
 - **ResourceMatchers[].version:** (*Optional*) version de la ressource à filtrer.

- **ResourceMatchers[].names:** (*Optional*) noms dans le champ Kubernetes metadata.name de la ressource à filtrer.
- **ResourceMatchers[].namespaces:** (*Optional*) Namespaces dans le champ Kubernetes metadata.name de la ressource à filtrer.
- **ResourceMatchers[].labelSelectors:** (*Optional*) chaîne de sélecteur de libellé dans le champ Kubernetes metadata.name de la ressource, comme défini dans le ["Documentation Kubernetes"](#). Par exemple : "trident.netapp.io/os=linux".

Par exemple :

```
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. Une fois que vous avez rempli le trident-protect-snapshot-restore-cr.yaml fichier avec les valeurs correctes, appliquez la CR :

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

Restaurez le snapshot à l'aide de l'interface de ligne de commande

Étapes

1. Restaurez l'instantané dans un autre espace de noms, en remplaçant les valeurs entre parenthèses par les informations de votre environnement.
 - L'snapshot`argument utilise un nom d'espace de noms et un nom d'instantané au format ``<namespace>/<name>`.
 - L'namespace-mapping`argument utilise des espaces de noms séparés par deux-points pour mapper les espaces de noms source aux espaces de noms de destination corrects dans le format ``source1:dest1,source2:dest2`.

Par exemple :

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

Gérer les crochets d'exécution de Trident Protect

Un crochet d'exécution est une action personnalisée que vous pouvez configurer pour s'exécuter conjointement avec une opération de protection des données d'une application gérée. Par exemple, si vous disposez d'une application de base de données, vous pouvez utiliser un crochet d'exécution pour suspendre toutes les transactions de base de données avant un instantané et reprendre les transactions une fois l'instantané terminé. Les snapshots sont ainsi cohérents au niveau des applications.

Types de crochets d'exécution

Trident Protect prend en charge les types de crochets d'exécution suivants, en fonction du moment où ils peuvent être exécutés :

- Pré-instantané
- Post-snapshot
- Avant sauvegarde
- Post-sauvegarde
- Post-restauration
- Après le basculement

Ordre d'exécution

Lors de l'exécution d'une opération de protection des données, les événements de hook d'exécution ont lieu dans l'ordre suivant :

1. Tous les crochets d'exécution de pré-opération personnalisés applicables sont exécutés sur les conteneurs appropriés. Vous pouvez créer et exécuter autant de crochets de pré-opération personnalisés que vous le souhaitez, mais l'ordre d'exécution de ces crochets avant que l'opération ne soit ni garantie ni configurable.
2. Le système de fichiers se bloque, le cas échéant. ["En savoir plus sur la configuration de la congélation du système de fichiers avec Trident Protect"](#).
3. L'opération de protection des données est effectuée.
4. Les systèmes de fichiers gelés ne sont pas gelés, le cas échéant.
5. Tous les crochets d'exécution de post-opération personnalisés applicables sont exécutés sur les conteneurs appropriés. Vous pouvez créer et exécuter autant de crochets post-opération personnalisés que vous le souhaitez, mais l'ordre d'exécution de ces crochets après l'opération n'est ni garanti ni configurable.

Si vous créez plusieurs crochets d'exécution du même type (par exemple, pré-instantané), l'ordre d'exécution de ces crochets n'est pas garanti. Cependant, l'ordre d'exécution des crochets de différents types est garanti.

Par exemple, voici l'ordre d'exécution d'une configuration qui a tous les types de crochets :

1. Crochets pré-instantanés exécutés
2. Crochets post-snapshot exécutés
3. Crochets de pré-secours exécutés
4. Crochets post-secours exécutés



L'exemple d'ordre précédent s'applique uniquement lorsque vous exécutez une sauvegarde qui n'utilise pas de snapshot existant.



Vous devez toujours tester vos scripts d'exécution avant de les activer dans un environnement de production. Vous pouvez utiliser la commande 'kubectl exec' pour tester aisément les scripts. Une fois que vous avez activé les crochets d'exécution dans un environnement de production, testez les snapshots et les sauvegardes obtenus pour vous assurer qu'ils sont cohérents. Pour ce faire, vous pouvez cloner l'application dans un espace de noms temporaire, restaurer le snapshot ou la sauvegarde, puis tester l'application.

Remarques importantes sur les crochets d'exécution personnalisés

Lors de la planification de crochets d'exécution pour vos applications, tenez compte des points suivants.

- Un crochet d'exécution doit utiliser un script pour effectuer des actions. De nombreux crochets d'exécution peuvent référencer le même script.
- Trident Protect exige que les scripts utilisés par les crochets d'exécution soient écrits au format de scripts shell exécutables.
- La taille du script est limitée à 96 Ko.
- Trident Protect utilise les paramètres du crochet d'exécution et les critères correspondants pour déterminer les crochets applicables à une opération de snapshot, de sauvegarde ou de restauration.



Puisque les crochets d'exécution réduisent ou désactivent complètement la fonctionnalité de l'application contre laquelle ils s'exécutent, vous devez toujours essayer de réduire le temps d'exécution de vos crochets personnalisés. Si vous démarrez une opération de sauvegarde ou d'instantané avec les crochets d'exécution associés, mais que vous l'annulez, les crochets sont toujours autorisés à s'exécuter si l'opération de sauvegarde ou d'instantané a déjà commencé. Cela signifie que la logique utilisée dans un crochet d'exécution post-sauvegarde ne peut pas présumer que la sauvegarde a été effectuée.

Filtres de crochet d'exécution

Lorsque vous ajoutez ou modifiez un crochet d'exécution pour une application, vous pouvez ajouter des filtres au crochet d'exécution pour gérer les conteneurs auxquels le crochet correspond. Les filtres sont utiles pour les applications qui utilisent la même image de conteneur sur tous les conteneurs, mais ils peuvent utiliser chaque image à des fins différentes (comme Elasticsearch). Les filtres vous permettent de créer des scénarios dans lesquels des crochets d'exécution s'exécutent sur certains conteneurs, mais pas nécessairement tous identiques. Si vous créez plusieurs filtres pour un seul crochet d'exécution, ils sont combinés avec un opérateur ET logique. Vous pouvez avoir jusqu'à 10 filtres actifs par crochet d'exécution.

Chaque filtre que vous ajoutez à un crochet d'exécution utilise une expression régulière pour faire correspondre les conteneurs de votre cluster. Lorsqu'un crochet correspond à un conteneur, le crochet exécute son script associé sur ce conteneur. Les expressions régulières pour les filtres utilisent la syntaxe de

l'expression régulière 2 (RE2), qui ne prend pas en charge la création d'un filtre qui exclut les conteneurs de la liste des correspondances. Pour plus d'informations sur la syntaxe prise en charge par Trident Protect pour les expressions régulières dans les filtres de hook d'exécution, reportez-vous à la section "[Prise en charge de la syntaxe de l'expression régulière 2 \(RE2\)](#)".



Si vous ajoutez un filtre d'espace de noms à un crochet d'exécution qui s'exécute après une opération de restauration ou de clonage et que la source et la destination de restauration ou de clonage sont dans des espaces de noms différents, le filtre d'espace de noms est appliqué uniquement à l'espace de noms de destination.

Exemples de crochet d'exécution

Visitez le "[Projet GitHub NetApp Verda](#)" pour télécharger des scripts d'exécution réels pour les applications courantes telles qu'Apache Cassandra et Elasticsearch. Vous pouvez également voir des exemples et obtenir des idées pour structurer vos propres crochets d'exécution personnalisés.

Créer un crochet d'exécution

Vous pouvez créer un crochet d'exécution personnalisé pour une application à l'aide de Trident Protect. Vous devez disposer d'autorisations propriétaire, administrateur ou membre pour créer des crochets d'exécution.

Utiliser une CR

Étapes

1. Créez le fichier de ressource personnalisée (CR) et nommez-le `trident-protect-hook.yaml`.
2. Configurez les attributs suivants en fonction de votre environnement Trident Protect et de la configuration du cluster :
 - **metadata.name:** (*required*) le nom de cette ressource personnalisée; choisissez un nom unique et sensible pour votre environnement.
 - **Spec.applicationRef:** (*required*) Nom Kubernetes de l'application pour laquelle exécuter le hook d'exécution.
 - **Spec.stage:** (*required*) Une chaîne indiquant quelle étape de l'action doit être exécutée par le crochet d'exécution. Valeurs possibles :
 - Pré
 - Post
 - **Spec.action:** (*required*) Une chaîne indiquant l'action que prendra le crochet d'exécution, en supposant que tous les filtres de crochet d'exécution spécifiés soient mis en correspondance. Valeurs possibles :
 - Snapshot
 - Sauvegarde
 - Restaurer
 - Basculement
 - **Spec.enabled:** (*Optional*) indique si ce hook d'exécution est activé ou désactivé. Si elle n'est pas spécifiée, la valeur par défaut est `true`.
 - **Spec.hookSource:** (*required*) chaîne contenant le script hook codé en base64.
 - **Spec.timeout:** (*Optional*) nombre définissant la durée en minutes pendant laquelle le crochet d'exécution est autorisé à s'exécuter. La valeur minimale est de 1 minute et la valeur par défaut est de 25 minutes si elle n'est pas spécifiée.
 - **Spec.arguments:** (*Optional*) liste YAML d'arguments que vous pouvez spécifier pour le crochet d'exécution.
 - **Spec.matchingCriteria:** (*Optional*) liste facultative de paires de valeurs de clé de critères, chaque paire constituant un filtre de crochet d'exécution. Vous pouvez ajouter jusqu'à 10 filtres par crochet d'exécution.
 - **Spec.matchingCriteria.type:** (*Optional*) chaîne identifiant le type de filtre du crochet d'exécution. Valeurs possibles :
 - ContainerImage
 - ContainerName
 - PodName
 - PodLabel
 - NamespaceName
 - **Spec.matchingCriteria.Value:** (*Optional*) Une chaîne ou Une expression régulière identifiant la valeur du filtre crochet d'exécution.

Exemple 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: IyEvYmluL2Jhc2gKZWNoYAiZXhhbXBsZSBzY3JpcHQiCg==
  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. Une fois que vous avez rempli le fichier CR avec les valeurs correctes, appliquez la CR :

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

Utilisez l'CLI

Étapes

1. Créez le crochet d'exécution en remplaçant les valeurs entre parenthèses par les informations de votre environnement. Par exemple :

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

Désinstallez Trident Protect

Vous devrez peut-être supprimer les composants Trident Protect si vous effectuez une mise à niveau d'une version d'évaluation vers une version complète du produit.

Pour supprimer Trident Protect, effectuez les opérations suivantes.

Étapes

1. Supprimez les fichiers Trident Protect CR :

```
helm uninstall -n trident-protect trident-protect-crds
```

2. Supprimer Trident Protect :

```
helm uninstall -n trident-protect trident-protect
```

3. Supprimez l'espace de noms Trident Protect :

```
kubectl delete ns trident-protect
```

Informations sur le copyright

Copyright © 2025 NetApp, Inc. Tous droits réservés. Imprimé aux États-Unis. Aucune partie de ce document protégé par copyright ne peut être reproduite sous quelque forme que ce soit ou selon quelque méthode que ce soit (graphique, électronique ou mécanique, notamment par photocopie, enregistrement ou stockage dans un système de récupération électronique) sans l'autorisation écrite préalable du détenteur du droit de copyright.

Les logiciels dérivés des éléments NetApp protégés par copyright sont soumis à la licence et à l'avis de non-responsabilité suivants :

CE LOGICIEL EST FOURNI PAR NETAPP « EN L'ÉTAT » ET SANS GARANTIES EXPRESSES OU TACITES, Y COMPRIS LES GARANTIES TACITES DE QUALITÉ MARCHANDE ET D'ADÉQUATION À UN USAGE PARTICULIER, QUI SONT EXCLUES PAR LES PRÉSENTES. EN AUCUN CAS NETAPP NE SERA TENU POUR RESPONSABLE DE DOMMAGES DIRECTS, INDIRECTS, ACCESSOIRES, PARTICULIERS OU EXEMPLAIRES (Y COMPRIS L'ACHAT DE BIENS ET DE SERVICES DE SUBSTITUTION, LA PERTE DE JOUISSANCE, DE DONNÉES OU DE PROFITS, OU L'INTERRUPTION D'ACTIVITÉ), QUELLES QU'EN SOIENT LA CAUSE ET LA DOCTRINE DE RESPONSABILITÉ, QU'IL S'AGISSE DE RESPONSABILITÉ CONTRACTUELLE, STRICTE OU DÉLICTELLE (Y COMPRIS LA NÉGLIGENCE OU AUTRE) DÉCOULANT DE L'UTILISATION DE CE LOGICIEL, MÊME SI LA SOCIÉTÉ A ÉTÉ INFORMÉE DE LA POSSIBILITÉ DE TELS DOMMAGES.

NetApp se réserve le droit de modifier les produits décrits dans le présent document à tout moment et sans préavis. NetApp décline toute responsabilité découlant de l'utilisation des produits décrits dans le présent document, sauf accord explicite écrit de NetApp. L'utilisation ou l'achat de ce produit ne concède pas de licence dans le cadre de droits de brevet, de droits de marque commerciale ou de tout autre droit de propriété intellectuelle de NetApp.

Le produit décrit dans ce manuel peut être protégé par un ou plusieurs brevets américains, étrangers ou par une demande en attente.

LÉGENDE DE RESTRICTION DES DROITS : L'utilisation, la duplication ou la divulgation par le gouvernement sont sujettes aux restrictions énoncées dans le sous-paragraphe (b)(3) de la clause Rights in Technical Data-Noncommercial Items du DFARS 252.227-7013 (février 2014) et du FAR 52.227-19 (décembre 2007).

Les données contenues dans les présentes se rapportent à un produit et/ou service commercial (tel que défini par la clause FAR 2.101). Il s'agit de données propriétaires de NetApp, Inc. Toutes les données techniques et tous les logiciels fournis par NetApp en vertu du présent Accord sont à caractère commercial et ont été exclusivement développés à l'aide de fonds privés. Le gouvernement des États-Unis dispose d'une licence limitée irrévocable, non exclusive, non cessible, non transférable et mondiale. Cette licence lui permet d'utiliser uniquement les données relatives au contrat du gouvernement des États-Unis d'après lequel les données lui ont été fournies ou celles qui sont nécessaires à son exécution. Sauf dispositions contraires énoncées dans les présentes, l'utilisation, la divulgation, la reproduction, la modification, l'exécution, l'affichage des données sont interdits sans avoir obtenu le consentement écrit préalable de NetApp, Inc. Les droits de licences du Département de la Défense du gouvernement des États-Unis se limitent aux droits identifiés par la clause 252.227-7015(b) du DFARS (février 2014).

Informations sur les marques commerciales

NETAPP, le logo NETAPP et les marques citées sur le site <http://www.netapp.com/TM> sont des marques déposées ou des marques commerciales de NetApp, Inc. Les autres noms de marques et de produits sont des marques commerciales de leurs propriétaires respectifs.