



# Protégez les applications avec Trident Protect

Trident

NetApp

November 14, 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 . . . . .	5
Installez le plug-in Trident Protect CLI . . . . .	9
Personnaliser l'installation de Trident Protect . . . . .	13
Gérez Trident Protect . . . . .	18
Gestion des autorisations et du contrôle d'accès Trident Protect . . . . .	18
Contrôlez les ressources Trident Protect . . . . .	25
Générez un pack de support Trident Protect . . . . .	30
Mettez à niveau Trident Protect . . . . .	32
Gérez et protégez les applications . . . . .	34
Utilisez les objets Trident Protect AppVault pour gérer les compartiments . . . . .	34
Définissez une application de gestion avec Trident Protect . . . . .	48
Protégez les applications à l'aide de Trident Protect . . . . .	52
Restauration des applications . . . . .	64
Répliquez des applications à l'aide de NetApp SnapMirror et Trident Protect . . . . .	81
Migrez vos applications à l'aide de Trident Protect . . . . .	98
Gérer les crochets d'exécution de Trident Protect . . . . .	102
Désinstallez Trident Protect . . . . .	114

# 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



- Les sauvegardes Trident Protect sont prises en charge uniquement sur les nœuds de calcul Linux. Les nœuds de calcul Windows ne sont pas pris en charge pour les opérations de sauvegarde.
- 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 Trident 24.02 ou une version plus récente (Trident 24.10 est recommandé).
- 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 offre des fonctionnalités de gel et de dégel du système de fichiers pour les machines virtuelles KubeVirt lors des opérations de protection des données afin de garantir la cohérence des données. La méthode de configuration et le comportement par défaut des opérations de gel de VM varient selon les versions de Trident Protect, les versions plus récentes offrant une configuration simplifiée via les paramètres du graphique Helm.



Pendant les opérations de restauration, tout `VirtualMachineS snapshots` créés pour une machine virtuelle (VM) ne sont pas restaurés.

### Trident Protect 25.10 et versions ultérieures

Trident Protect gèle et débloque automatiquement les systèmes de fichiers KubeVirt pendant les opérations de protection des données afin de garantir la cohérence. À partir de Trident Protect 25.10, vous pouvez désactiver ce comportement à l'aide de `vm.freeze` paramètre lors de l'installation du graphique Helm. Ce paramètre est activé par défaut.

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

### Trident protège du 24.10.1 au 25.06

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
```

## 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
```

## Conditions requises pour la réplication SnapMirror

La réplication NetApp SnapMirror est disponible pour Trident Protect dans les solutions ONTAP suivantes :

- Clusters NetApp FAS, AFF et ASA sur site
- 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 :

- **NetApp Trident** : NetApp Trident doit exister sur les clusters Kubernetes source et de destination qui utilisent ONTAP comme backend. 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 : NFS
  - ontap-san : iSCSI
  - ontap-san : FC
  - ontap-san : NVMe/TCP (nécessite au minimum la version ONTAP 9.15.1)
- **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](#)" .

À partir de ONTAP 9.10.1, toutes les licences sont livrées sous forme de fichier de licence NetApp (NLF), qui est un fichier unique qui active plusieurs fonctionnalités. Pour plus d'informations, reportez-vous à la section "[Licences incluses avec ONTAP One](#)" .



Seule la protection asynchrone SnapMirror est prise en charge.

## 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.

- **NetApp Trident et SVM** : les SVM distantes appairées doivent être disponibles pour NetApp Trident sur le cluster de 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.

## 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.

## Exigences du cluster Kubernetes pour la réPLICATION SnapMirror

Assurez-vous que vos clusters Kubernetes répondent aux exigences suivantes :

- **Accessibilité AppVault** : les clusters source et de destination doivent disposer d'un accès réseau pour lire et écrire dans AppVault pour la réPLICATION des objets d'application.
- **Connectivité réseau** : configurez les règles de pare-feu, les autorisations de compartiment et les listes d'adresses IP autorisées pour permettre la communication entre les deux clusters et AppVault sur les réseaux WAN.



De nombreux environnements d'entreprise mettent en œuvre des politiques de pare-feu strictes sur les connexions WAN. Vérifiez ces exigences réseau avec votre équipe d'infrastructure avant de configurer la réPLICATION.

## 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.

### 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. Utilisez Helm pour installer Trident Protect. Remplacer <name-of-cluster> par un nom de cluster, qui sera attribué au cluster et utilisé pour identifier les sauvegardes et snapshots du cluster :

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

3. Pour activer la journalisation de débogage (recommandée pour le dépannage), vous pouvez utiliser la commande suivante (facultative) :

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

La journalisation de débogage aide le support NetApp à résoudre les problèmes sans nécessiter de modifications du niveau de journalisation ni de reproduction du problème.

## 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 :

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

Par exemple :

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

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

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



Pour obtenir la carte de barre, commencez par la télécharger sur un ordinateur connecté à Internet. `helm pull trident-protect --version 100.2510.0 --repo https://netapp.github.io/trident-protect-helm-chart`, puis copiez le résultat `trident-protect-100.2510.0.tgz`. Téléchargez le fichier dans votre environnement hors ligne et installez-le en utilisant `helm install trident-protect ./trident-protect-100.2510.0.tgz` au lieu de la référence au dépôt dans l'étape finale.

2. Créez 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 :

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



Le `imageRegistry` et `imagePullSecrets` Les valeurs s'appliquent à toutes les images des composants, y compris `resourcebackup` et `resourcerestore`. Si vous envoyez des images vers un chemin de dépôt spécifique au sein de votre registre (par exemple, `example.com:443/my-repo`), veuillez inclure le chemin complet dans le champ d'enregistrement. Cela permettra de garantir que toutes les images sont extraites de `<private-registry-url>/<image-name>:<tag>`.

##### 7. Utilisez Helm pour installer Trident Protect. Remplacer `<name_of_cluster>` par un nom de cluster, qui sera attribué au cluster et utilisé pour identifier les sauvegardes et snapshots du cluster :

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

##### 8. Pour activer la journalisation de débogage (recommandée pour le dépannage), vous pouvez utiliser la commande suivante (facultative) :

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

La journalisation de débogage aide le support NetApp à résoudre les problèmes sans nécessiter de modifications du niveau de journalisation ni de reproduction du problème.



Pour plus d'options de configuration des graphiques Helm, notamment les paramètres AutoSupport et le filtrage des espaces de noms, veuillez consulter la documentation. ["Personnaliser l'installation de Trident Protect"](#).

## 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/25.10.0/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/25.10.0/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/25.10.0/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/25.10.0/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 Privileges 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. Créez le script de compléction :

```
tridentctl-protect completion bash > 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. Créez le script de compléction :

```
tridentctl-protect completion zsh > 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.

## Personnaliser l'installation de Trident Protect

Vous pouvez personnaliser la configuration par défaut de Trident Protect en fonction des besoins spécifiques de votre environnement.

### 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 netapp-trident-protect/trident-protect -f resourceLimits.yaml --reuse-values
```

## Personnaliser les contraintes de contexte de sécurité

Vous pouvez utiliser un fichier de configuration pour modifier les restrictions de contexte de sécurité OpenShift pour les conteneurs Trident Protect après avoir installé Trident Protect. Ces contraintes définissent des restrictions de sécurité pour les pods d'un cluster Red Hat OpenShift.

### Étapes

1. Créez un fichier nommé `sccconfig.yaml`.
2. Ajoutez l'option SCC au fichier et modifiez les paramètres en fonction des besoins de votre environnement.

L'exemple suivant montre les valeurs par défaut des paramètres de l'option SCC :

```

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

```

Ce tableau décrit les paramètres de l'option SCC :

Paramètre	Description	Valeur par défaut
création	Détermine si une ressource SCC peut être créée. Une ressource SCC ne sera créée que si est défini sur <code>true</code> et que <code>scc.create</code> le processus d'installation de Helm identifie un environnement OpenShift. Si ne fonctionne pas sur OpenShift ou si <code>scc.create</code> est défini sur <code>false</code> , aucune ressource SCC ne sera créée.	vrai
nom	Spécifie le nom du SCC.	travail-protection-Trident
priorité	Définit la priorité du SCC. Les SCC ayant des valeurs de priorité plus élevées sont évalués avant ceux ayant des valeurs plus faibles.	1

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

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

Les valeurs par défaut seront remplacées par celles spécifiées dans le `sccconfig.yaml` fichier.

### Configurer des paramètres supplémentaires du graphique de protection Trident

Vous pouvez personnaliser les paramètres AutoSupport et le filtrage des espaces de noms pour répondre à vos besoins spécifiques. Le tableau suivant décrit les paramètres de configuration disponibles :

Paramètre	Type	Description
<code>autoSupport.proxy</code>	chaîne	Configure une URL proxy pour les connexions NetApp AutoSupport. Utilisez ceci pour acheminer les téléchargements de bundles de support via un serveur proxy. Exemple: <a href="http://my.proxy.url">http://my.proxy.url</a> .
<code>autoSupport.insecure</code>	booléen	Ignore la vérification TLS pour les connexions proxy AutoSupport lorsqu'elle est définie sur <code>true</code> . À utiliser uniquement pour les connexions proxy non sécurisées. (défaut: <code>false</code> )

Paramètre	Type	Description
autoSupport.activated	booléen	Active ou désactive les téléchargements quotidiens du bundle AutoSupport de protection Trident . Lorsqu'il est réglé sur <code>false</code> , les téléchargements quotidiens programmés sont désactivés, mais vous pouvez toujours générer manuellement des lots d'assistance. (défaut: <code>true</code> )
restaurerSkipNamespaceAnnotations	chaîne	Liste séparée par des virgules d'annotations d'espace de noms à exclure des opérations de sauvegarde et de restauration. Vous permet de filtrer les espaces de noms en fonction des annotations.
restaurerSkipNamespaceLabels	chaîne	Liste séparée par des virgules d'étiquettes d'espace de noms à exclure des opérations de sauvegarde et de restauration. Vous permet de filtrer les espaces de noms en fonction des étiquettes.

Vous pouvez configurer ces options à l'aide d'un fichier de configuration YAML ou d'indicateurs de ligne de commande :

## Utiliser le fichier YAML

### Étapes

1. Créez un fichier de configuration et nommez-le `values.yaml`.
2. Dans le fichier que vous avez créé, ajoutez les options de configuration que vous souhaitez personnaliser.

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

3. Après avoir rempli le `values.yaml` fichier avec les valeurs correctes, appliquez le fichier de configuration :

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

## Utiliser l'indicateur CLI

### Étapes

1. Utilisez la commande suivante avec le `--set` indicateur pour spécifier des paramètres individuels :

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

## Limitez les pods de protection Trident à des nœuds spécifiques

Vous pouvez utiliser la contrainte de sélection de nœud du sélecteur de nœuds Kubernetes pour déterminer lequel de vos nœuds peut exécuter les pods de protection Trident, en fonction des étiquettes de nœuds. Par défaut, Trident Protect est limité aux nœuds qui exécutent Linux. Vous pouvez personnaliser davantage ces contraintes en fonction de vos besoins.

### Étapes

1. Créez un fichier nommé `nodeSelectorConfig.yaml`.
2. Ajoutez l'option `nodeSelector` au fichier et modifiez le fichier pour ajouter ou modifier des libellés de nœud

afin de les restreindre en fonction des besoins de votre environnement. Par exemple, le fichier suivant contient la restriction par défaut du système d'exploitation, mais cible également une région et un nom d'application spécifiques :

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

3. Appliquer les valeurs du nodeSelectorConfig.yaml fichier :

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

Ceci remplace les restrictions par défaut par celles que vous avez spécifiées dans le nodeSelectorConfig.yaml fichier.

## 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
```

- 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

- 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
```

- 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

```

- 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.

## Contrôlez les ressources Trident Protect

Vous pouvez utiliser les metrics kube-state, Prometheus et les outils open source d'AlertManager pour contrôler l'état des ressources protégées par Trident Protect.

Le service de metrics kube-state génère des metrics à partir des communications de l'API Kubernetes. En l'utilisant avec Trident Protect, vous pourrez consulter des informations utiles sur l'état des ressources de votre environnement.

Prometheus est un kit d'outils qui permet d'ingérer les données générées par des metrics kube-state et de les présenter comme des informations faciles à lire sur ces objets. Ensemble, les metrics kube-état et Prometheus vous permettent de contrôler l'état et l'état des ressources que vous gérez avec Trident Protect.

AlertManager est un service qui ingère les alertes envoyées par des outils tels que Prometheus et les redirige vers les destinations que vous configurez.

Les configurations et les conseils inclus dans ces étapes ne sont que des exemples ; vous devez les personnaliser en fonction de votre environnement. Reportez-vous à la documentation officielle suivante pour obtenir des instructions et une assistance spécifiques :



- ["documentation sur les indicateurs d'état kube"](#)
- ["Documentation Prometheus"](#)
- ["Documentation d'AlertManager"](#)

## Étape 1 : installez les outils de surveillance

Pour activer la surveillance des ressources dans Trident Protect, vous devez installer et configurer kube-state-metrics, Prometheus et AlertManager.

### Installez les indicateurs kube-state

Vous pouvez installer kube-state-metrics à l'aide de Helm.

#### Étapes

1. Ajoutez le graphique Helm kube-state-metrics. Par exemple :

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

2. Appliquez le CRD Prometheus ServiceMonitor au cluster :

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

3. Créez un fichier de configuration pour le graphique Helm (par exemple, `metrics-config.yaml`). Vous pouvez personnaliser l'exemple de configuration suivant en fonction de votre environnement :

## Metrics-config.yaml : configuration du graphique Helm kube-state-metrics

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

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

# Collect metrics from all namespaces
namespaces: ""

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

4. Installez les indicateurs d'état kube en déployant le graphique Helm. Par exemple :

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

5. Configurez les indicateurs d'état kube pour générer des mesures pour les ressources personnalisées utilisées par Trident Protect en suivant les instructions de la ["documentation sur les ressources personnalisées kube-state-metrics"](#).

#### Installez Prometheus

Vous pouvez installer Prometheus en suivant les instructions de la ["Documentation Prometheus"](#).

#### Installez AlertManager

Vous pouvez installer AlertManager en suivant les instructions de la ["Documentation d'AlertManager"](#).

### Étape 2 : configurer les outils de surveillance pour qu'ils fonctionnent ensemble

Après avoir installé les outils de surveillance, vous devez les configurer pour qu'ils fonctionnent ensemble.

#### Étapes

1. Intégrez des metrics kube-state avec Prometheus. Modifiez le fichier de configuration Prometheus (prometheus.yaml) et ajoutez les informations du service kube-state-metrics. Par exemple :

#### prometheus.yaml : intégration du service kube-state-metrics avec Prometheus

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

2. Configurez Prometheus pour acheminer les alertes vers AlertManager. Modifiez le fichier de configuration Prometheus (prometheus.yaml) et ajoutez la section suivante :

## **prometheus.yaml : envoyer des alertes à Alertmanager**

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

### **Résultat**

Prometheus peut désormais collecter des metrics à partir de metrics kube-state et envoyer des alertes à AlertManager. Vous êtes maintenant prêt à configurer les conditions qui déclenchent une alerte et l'emplacement où les alertes doivent être envoyées.

### **Étape 3 : configuration des alertes et des destinations d'alertes**

Une fois que vous avez configuré les outils pour qu'ils fonctionnent ensemble, vous devez configurer le type d'informations qui déclenche des alertes et l'emplacement où elles doivent être envoyées.

#### **Exemple d'alerte : échec de la sauvegarde**

L'exemple suivant définit une alerte critique qui est déclenchée lorsque l'état de la ressource personnalisée de sauvegarde est défini sur Error pendant 5 secondes ou plus. Vous pouvez personnaliser cet exemple pour l'adapter à votre environnement et inclure cet extrait YAML dans votre `prometheus.yaml` fichier de configuration :

## **rules.yaml : définir une alerte Prometheus pour les sauvegardes ayant échoué**

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

### **Configurez AlertManager pour envoyer des alertes à d'autres canaux**

Vous pouvez configurer AlertManager pour envoyer des notifications à d'autres canaux, tels que les e-mails, PagerDuty, Microsoft Teams ou d'autres services de notification en spécifiant la configuration respective dans le `alertmanager.yaml` fichier.

L'exemple suivant configure AlertManager pour envoyer des notifications à un canal Slack. Pour personnaliser cet exemple en fonction de votre environnement, remplacez la valeur de la `api_url` clé par l'URL Slack webhook utilisée dans votre environnement :

## alertmanager.yaml : envoyer des alertes à un canal Slack

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

## Générez un pack de support Trident Protect

Trident Protect permet aux administrateurs de générer des bundles contenant des informations utiles au support NetApp , notamment des journaux, des mesures et des informations de topologie sur les clusters et les applications sous gestion. Si vous êtes connecté à Internet, vous pouvez télécharger des bundles de support sur le site de support NetApp (NSS) à l'aide d'un fichier de ressources personnalisé (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. Après avoir rempli le `trident-protect-support-bundle.yaml` fichier avec les valeurs correctes, appliquez le CR :

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

## 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> -n trident-protect
```

## Surveiller et récupérer le pack de support

Après avoir créé un bundle de support à l'aide de l'une ou l'autre méthode, vous pouvez surveiller sa progression de génération et le récupérer sur votre système local.

### Étapes

1. Attendez le `status.generationState` atteindre `Completed` État. Vous pouvez surveiller la progression de la génération avec la commande suivante :

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

2. Récupérez le pack de support sur votre système local. Obtenez la commande de copie à partir du bundle AutoSupport terminé :

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

Trouver le `kubectl cp` commande à partir de la sortie et exécutez-la, en remplaçant l'argument de destination par votre répertoire local préféré.

## 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.

- Lors d'une mise à niveau depuis la version 24.10, les snapshots exécutés pendant la mise à niveau peuvent échouer. Cet échec n'empêche pas la création de futurs snapshots, qu'ils soient manuels ou planifiés. Si un snapshot échoue pendant la mise à niveau, vous pouvez en créer un manuellement pour garantir la protection de votre application.



Pour éviter d'éventuels échecs, vous pouvez désactiver toutes les planifications d'instantanés avant la mise à niveau et les réactiver ensuite. Cependant, cela entraînera l'absence d'instantanés planifiés pendant la période de mise à niveau.

- Pour les installations de registre privé, assurez-vous que le graphique Helm et les images requis pour la version cible sont disponibles dans votre registre privé et vérifiez que vos valeurs Helm personnalisées sont compatibles avec la nouvelle version du graphique. Pour plus d'informations, veuillez consulter "[Installez Trident Protect à partir d'un registre privé](#)".

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 :



Cette étape est requise si vous effectuez une mise à niveau à partir d'une version antérieure à 25.06, car les CRD sont désormais inclus dans le tableau de protection du casque Trident.

- a. Exécutez cette commande pour déplacer la gestion des CRD de `trident-protect-crds` à `trident-protect` :

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

- b. Exécutez cette commande pour supprimer le secret Helm pour le `trident-protect-crds` graphique :



Ne désinstallez pas le `trident-protect-crds` graphique à l'aide de Helm, car cela pourrait supprimer vos CRD et toutes les données associées.

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

3. Mise à niveau de Trident Protect :

```
helm upgrade trident-protect netapp-trident-protect/trident-protect  
--version 100.2510.0 --namespace trident-protect
```



Vous pouvez configurer le niveau de journalisation lors de la mise à niveau en ajoutant `--set logLevel=debug` à la commande de mise à niveau. Le niveau de journalisation par défaut est `warn`. L'activation de la journalisation de débogage est recommandée pour le dépannage, car elle aide le support NetApp à diagnostiquer les problèmes sans nécessiter de modifications du niveau de journalisation ni de reproduction du problème.

## 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.

Vous devez créer une CR AppVault manuellement ou depuis la ligne de commande lorsque vous effectuez des opérations de protection des données sur une application. La CR AppVault est spécifique à votre environnement et vous pouvez vous inspirer des exemples de cette page pour créer des CR AppVault.



Assurez-vous que le CR AppVault se trouve sur le cluster où Trident Protect est installé. Si le CR AppVault n'existe pas ou si vous ne pouvez pas y accéder, la ligne de commande affiche une erreur.

### Configurer l'authentification et les mots de passe AppVault

Avant de créer un AppVault CR, assurez-vous que l'AppVault et le dispositif de transfert de données que vous choisissez peuvent s'authentifier auprès du fournisseur et de toutes les ressources associées.

#### Mots de passe du référentiel de Data Mover

Lorsque vous créez des objets AppVault à l'aide de CR ou du plug-in CLI Trident Protect, vous pouvez spécifier un secret Kubernetes avec des mots de passe personnalisés pour le chiffrement Restic et Kopia. Si vous ne spécifiez pas de secret, Trident Protect utilise un mot de passe par défaut.

- Lors de la création manuelle de CR AppVault, utilisez le champ `spec.dataMoverPasswordSecretRef` pour spécifier le secret.
- Lors de la création d'objets AppVault à l'aide de l'interface de ligne de commande Trident Protect, utilisez le `--data-mover-password-secret-ref` argument pour spécifier le secret.

#### Créez un mot de passe secret pour le référentiel du Data Mover

Utilisez les exemples suivants pour créer le mot de passe secret. Lorsque vous créez des objets AppVault, vous pouvez demander à Trident Protect d'utiliser ce secret pour s'authentifier auprès du référentiel Data

- Selon le mécanisme de déplacement des données que vous utilisez, il vous suffit d'inclure le mot de passe correspondant à ce mécanisme de transfert de données. Par exemple, si vous utilisez Restic et que vous ne prévoyez pas d'utiliser Kopia à l'avenir, vous ne pouvez inclure que le mot de passe Restic lorsque vous créez le secret.
- Conservez le mot de passe en lieu sûr. Vous en aurez besoin pour restaurer les données sur le même cluster ou sur un autre. Si le cluster ou le trident-protect l'espace de noms est supprimé, vous ne pourrez pas restaurer vos sauvegardes ou instantanés sans le mot de passe.

### Utiliser une CR

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

### Utiliser l'interface de ligne de commande

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

### Autorisations IAM de stockage compatibles S3

Lorsque vous accédez à un stockage compatible S3 tel qu'Amazon S3, Generic S3, "StorageGRID S3", ou "ONTAP S3" Avec Trident Protect, vous devez vous assurer que les identifiants utilisateur fournis disposent des autorisations nécessaires pour accéder au bucket. Voici un exemple de politique accordant les autorisations minimales requises pour l'accès avec Trident Protect. Vous pouvez appliquer cette politique à l'utilisateur qui gère les politiques de bucket compatibles S3.

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

Pour plus d'informations sur les politiques Amazon S3, reportez-vous aux exemples dans le ["Documentation Amazon S3"](#).

#### **Identité du pod EKS pour l'authentification Amazon S3 (AWS)**

Trident Protect prend en charge EKS Pod Identity pour les opérations de transfert de données Kopia. Cette fonctionnalité permet un accès sécurisé aux buckets S3 sans stocker les informations d'identification AWS dans les secrets Kubernetes.

\*Exigences pour l'identité du pod EKS avec protection Trident \*

Avant d'utiliser EKS Pod Identity avec Trident Protect, assurez-vous des points suivants :

- L'identité de pod est activée sur votre cluster EKS.
- Vous avez créé un rôle IAM avec les autorisations de compartiment S3 nécessaires. Pour en savoir plus, consultez ["Autorisations IAM de stockage compatibles S3"](#).
- Le rôle IAM est associé aux comptes de service de protection Trident suivants :
  - <trident-protect>-controller-manager
  - <trident-protect>-resource-backup
  - <trident-protect>-resource-restore
  - <trident-protect>-resource-delete

Pour obtenir des instructions détaillées sur l'activation de l'identité du pod et l'association des rôles IAM aux comptes de service, reportez-vous à la ["Documentation sur l'identité des pods AWS EKS"](#).

**Configuration AppVault** Lorsque vous utilisez l'identité du pod EKS, configurez votre AppVault CR avec le `useIAM: true` drapeau au lieu d'informations d'identification explicites :

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

#### Exemples de génération de clés AppVault pour les fournisseurs cloud

Lors de la définition d'un AppVault CR, vous devez inclure des informations d'identification pour accéder aux ressources hébergées par le fournisseur, sauf si vous utilisez l'authentification IAM. La manière dont vous générez les clés pour les informations d'identification varie selon le fournisseur. Voici des exemples de génération de clés en ligne de commande pour plusieurs fournisseurs. Vous pouvez utiliser les exemples suivants pour créer des clés pour les informations d'identification de chaque fournisseur de 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=<amazon-s3-trident-protect-src-bucket
-secret> \
-n trident-protect
```

## Microsoft Azure

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

## S3 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=<ontap-s3-trident-protect-src-bucket
-secret> \
-n trident-protect
```

## StorageGRID S3

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

## Exemples de création d'AppVault

Voici des exemples de définitions AppVault pour chaque fournisseur.

### Exemples de CR AppVault

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

- Vous pouvez facultativement spécifier un code secret Kubernetes qui contient des mots de passe personnalisés pour le chiffrement du référentiel Restic et Kopia. Pour plus d'informations, reportez-vous à la section [Mots de passe du référentiel de Data Mover](#) .
- Pour les objets AppVault Amazon S3 (AWS), vous pouvez spécifier un jeton de session, utile si vous utilisez l'authentification SSO. Ce jeton est créé lorsque vous générez des clés pour le fournisseur dans [Exemples de génération de clés AppVault pour les fournisseurs cloud](#).
- Pour les objets S3 AppVault, vous pouvez spécifier une URL proxy de sortie pour le trafic S3 sortant à l'aide de la `spec.providerConfig.S3.proxyURL` clé.



## Google Cloud

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

## Amazon S3 (AWS)

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



Pour les environnements EKS utilisant Pod Identity avec le déplaceur de données Kopia, vous pouvez supprimer la section providerCredentials et ajouter useIAM: true sous la configuration s3.

## Microsoft Azure

```

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

```

## S3 générique

```

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

```

## ONTAP S3

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: ontap-s3-trident-protect-src-bucket
  namespace: trident-protect
spec:
  dataMoverPasswordSecretRef: my-optional-data-mover-secret
  providerType: Ontaps3
  providerConfig:
    s3:
      bucketName: trident-protect-src-bucket
      endpoint: s3.example.com
      proxyURL: http://10.1.1.1:3128
  providerCredentials:
    accessKeyID:
      valueFromSecret:
        key: accessKeyID
        name: s3-secret
    secretAccessKey:
      valueFromSecret:
        key: secretAccessKey
        name: s3-secret
```

## StorageGRID S3

```

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

```

#### 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.

- Vous pouvez facultativement spécifier un code secret Kubernetes qui contient des mots de passe personnalisés pour le chiffrement du référentiel Restic et Kopia. Pour plus d'informations, reportez-vous à la section [Mots de passe du référentiel de Data Mover](#) .
- Pour les objets S3 AppVault, vous pouvez spécifier une URL de sortie proxy pour le trafic S3 sortant à l'aide de l' `--proxy-url <ip_address:port>` argument.

## Google Cloud

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

## Amazon S3 (AWS)

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

## Microsoft Azure

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

## S3 générique

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

## ONTAP S3

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

## StorageGRID S3

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

## Afficher les informations AppVault

Vous pouvez utiliser le plug-in Trident Protect CLI pour afficher les informations sur les objets AppVault que vous avez créés sur le cluster.

### Étapes

1. Afficher le contenu d'un objet AppVault :

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

**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)

- 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 CRS de snapshot et de sauvegarde utilisés par l'AppVault que vous souhaitez supprimer.

## 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 l'espace de noms et le sélecteur d'étiquettes pour spécifier les espaces de noms et les ressources utilisés par l'application. L'espace de nom de l'application doit faire partie de cette liste. Le sélecteur d'étiquettes est facultatif et peut être utilisé pour filtrer les ressources dans chaque espace de noms spécifié.
  - **spec.includedClusterScopedResources**: (*Optional*) utilisez cet attribut pour spécifier les ressources cluster-scoped à inclure dans la définition de l'application. Cet attribut vous permet de sélectionner ces ressources en fonction de leur groupe, de leur version, de leur type et de leurs étiquettes.
    - **GroupVersionKind**: (*required*) Spécifie le groupe d'API, la version et le type de la ressource cluster-scoped.
    - **LabelSelector**: (*Optional*) filtre les ressources du cluster-scoped en fonction de leurs étiquettes.
  - **metadata.annotations.protect.trident.netapp.io/skip-vm-freeze**: (*Optional*) cette annotation s'applique uniquement aux applications définies à partir de machines virtuelles, comme dans les environnements KubeVirt, où les fichiers se figent avant les snapshots. Spécifiez si cette application peut écrire dans le système de fichiers pendant un snapshot. Si elle est définie sur `true`, l'application ignore le paramètre global et peut écrire dans le système de fichiers pendant un snapshot. Si elle est définie sur `FALSE`, l'application ignore le paramètre global et le système de fichiers est bloqué pendant un snapshot. Si elle est spécifiée mais que l'application ne comporte aucune machine virtuelle dans la définition de l'application, l'annotation est ignorée. S'il n'est pas spécifié, l'application suit le "[Paramètre de gel Global Trident Protect](#)".

Si vous devez appliquer cette annotation après la création d'une application, vous pouvez utiliser la commande suivante :

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

+

Exemple YAML :

+

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

1. (Facultatif) 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".



Quand les deux resourceFilter et labelSelector sont utilisés, resourceFilter court d'abord, puis labelSelector est appliqué aux ressources résultantes.

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"]
```

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
```

## Étapes

1. Créez et appliquez la définition de l'application à l'aide de l'un des exemples suivants, 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 de l'application à l'aide de listes séparées par des virgules avec les arguments présentés dans les exemples.

Vous pouvez éventuellement utiliser une annotation lorsque vous créez une application pour spécifier si l'application peut écrire dans le système de fichiers pendant un snapshot. Cela s'applique uniquement aux applications définies à partir de machines virtuelles, par exemple dans les environnements KubeVirt, où les systèmes de fichiers se figent avant les snapshots. Si vous définissez l'annotation sur `true`, l'application ignore le paramètre `global` et peut écrire dans le système de fichiers pendant un snapshot. Si vous le définissez sur `false`, l'application ignore le

paramètre global et le système de fichiers est bloqué pendant un instantané. Si vous utilisez l'annotation mais que l'application n'a pas de machines virtuelles dans la définition de l'application, l'annotation est ignorée. Si vous n'utilisez pas l'annotation, l'application suit le "[Paramètre de gel Global Trident Protect](#)".

Pour spécifier l'annotation lorsque vous créez une application à l'aide de l'interface de ligne de commande, vous pouvez utiliser `--annotation` l'indicateur.

- Créez l'application et utilisez le paramètre global pour le comportement de blocage du système de fichiers :

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

- Créez l'application et configurez le paramètre de l'application locale pour le comportement de blocage du système de fichiers :

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

Vous pouvez utiliser `--resource-filter-include` et `--resource-filter-exclude` indicateurs pour inclure ou exclure des ressources en fonction de `resourceSelectionCriteria` tels que le groupe, le type, la version, les étiquettes, les noms et les espaces de noms, comme illustré dans l'exemple suivant :

```
tridentctl-protect create application <my_new_app_cr_name>
--namespaces <namespaces_to_include> --csr
<clusterScopedResources_to_include> --namespace <my-app-namespace>
--resource-filter-include
' [ {"Group": "apps", "Kind": "Deployment", "Version": "v1", "Names": [ "my-
deployment" ], "Namespaces": [ "my-
namespace" ], "LabelSelectors": [ "app=my-app" ] } ] '
```

## 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.



Les ressources Cluster-scoped sont incluses dans une sauvegarde, un snapshot ou un clone s'ils sont explicitement référencés dans la définition d'application ou s'ils ont des références à l'un des namespaces d'application.

## 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.



Les ressources Cluster-scoped sont incluses dans une sauvegarde, un snapshot ou un clone s'ils sont explicitement référencés dans la définition d'application ou s'ils ont des références à l'un des namespaces d'application.

### Avant de commencer

Assurez-vous que l'expiration du jeton de session AWS suffit pour toutes les opérations de sauvegarde s3 à long terme. Si le jeton expire pendant l'opération de sauvegarde, l'opération peut échouer.

- Pour plus d'informations sur la vérification de l'expiration du jeton de session en cours, reportez-vous "[Documentation de l'API AWS](#)" au.
- Pour plus d'informations sur les identifiants avec les ressources AWS, consultez le "[Documentation AWS IAM](#)".

## 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.

Exemple YAML :

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

3. 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

- 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> --data-mover <Kopia_or_Restic> -n <application_namespace>
```

Vous pouvez éventuellement utiliser --full-backup l'indicateur pour spécifier si une sauvegarde doit être non incrémentielle. Par défaut, toutes les sauvegardes sont incrémentielles. Lorsque cette option est utilisée, la sauvegarde devient non incrémentielle. Il est recommandé d'effectuer régulièrement une sauvegarde complète, puis d'effectuer des sauvegardes incrémentielles entre deux sauvegardes complètes afin de réduire le risque associé aux restaurations.

#### Annotations de sauvegarde prises en charge

Le tableau suivant décrit les annotations que vous pouvez utiliser lors de la création d'une demande de modification de sauvegarde :

Annotation	Type	Description	Valeur par défaut
protect.trident.netapp.io/full-backup	chaîne	Indique si la sauvegarde doit être non incrémentale. Réglé sur true créer une sauvegarde non incrémentale. Il est recommandé d'effectuer périodiquement une sauvegarde complète, puis d'effectuer des sauvegardes incrémentielles entre les sauvegardes complètes afin de minimiser les risques liés aux restaurations.	« faux »
protect.trident.netapp.io/snaps-hot-completion-timeout	chaîne	Durée maximale autorisée pour l'exécution de l'opération globale de capture d'instantané.	"60m"
protect.trident.netapp.io/volume-snapshots-ready-to-use-timeout	chaîne	Le délai maximal autorisé pour que les instantanés de volume atteignent l'état prêt à l'emploi.	"30m"
protect.trident.netapp.io/volume-snapshots-created-timeout	chaîne	Durée maximale autorisée pour la création d'instantanés de volume.	"5m"
protect.trident.netapp.io/pvc-bind-timeout-sec	chaîne	Délai maximal (en secondes) d'attente pour que les PersistentVolumeClaims (PVC) nouvellement créées atteignent le Bound phase précédant l'échec des opérations.	"1200" (20 minutes)

#### Créez un calendrier de protection des données

Une politique de protection protège une application en créant des instantanés, des sauvegardes ou les deux selon un calendrier défini. Vous pouvez choisir de créer des instantanés 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. Vous pouvez planifier une sauvegarde complète non incrémentielle en utilisant l'annotation full-backup-rule. Par défaut, toutes les sauvegardes sont incrémentielles. L'exécution périodique d'une sauvegarde complète,

ainsi que de sauvegardes incrémentielles entre les deux, permet de réduire le risque associé aux restaurations.

- Vous pouvez créer des planifications pour les instantanés uniquement en définissant `backupRetention` à zéro et `snapshotRetention` à une valeur supérieure à zéro. Paramètre `snapshotRetention` à zéro signifie que toutes les sauvegardes planifiées créeront toujours des instantanés, mais ceux-ci sont temporaires et seront supprimés immédiatement après la fin de la sauvegarde.
- Les ressources Cluster-scoped sont incluses dans une sauvegarde, un snapshot ou un clone s'ils sont explicitement référencés dans la définition d'application ou s'ils ont des références à l'un des namespaces d'application.



## 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 (instantanés uniquement).
  - **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 )
    - Custom
  - **spec.dayOfMonth**: (*Facultatif*) Le jour du mois (1 - 31) auquel la planification doit s'exécuter. Ce champ est obligatoire si la granularité est définie sur `Monthly` . La valeur doit être fournie sous forme de chaîne.
  - **spec.dayOfWeek**: (*Facultatif*) Le jour de la semaine (0 - 7) pendant lequel la planification doit s'exécuter. Les valeurs de 0 ou 7 indiquent dimanche. Ce champ est obligatoire si la granularité est définie sur `Weekly` . La valeur doit être fournie sous forme de chaîne.
  - **spec.hour**: (*Facultatif*) L'heure de la journée (0 - 23) à laquelle le programme doit s'exécuter. Ce champ est obligatoire si la granularité est définie sur `Daily` , `Weekly` , ou `Monthly` . La valeur doit être fournie sous forme de chaîne.
  - **spec.minute**: (*Facultatif*) La minute de l'heure (0 - 59) à laquelle la planification doit s'exécuter. Ce champ est obligatoire si la granularité est définie sur `Hourly` , `Daily` , `Weekly` , ou `Monthly` . La valeur doit être fournie sous forme de chaîne.

Exemple de YAML pour la planification de sauvegarde et de snapshot :

```

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

```

Exemple de YAML pour la planification des instantanés uniquement :

```

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

```

3. 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>
--full-backup-rule <string>
```

Vous pouvez définir le `--full-backup-rule` drapeau sur `always` pour une sauvegarde complète constante ou le personnaliser en fonction de vos besoins. Par exemple, si vous choisissez la granularité quotidienne, vous pouvez spécifier les jours de la semaine sur lesquels la sauvegarde complète doit avoir lieu. Par exemple, utilisez `--full-backup-rule "Monday, Thursday"` pour programmer une sauvegarde complète le lundi et le jeudi.

Pour les planifications d'instantanés uniquement, définissez `--backup-retention 0` et spécifiez une valeur supérieure à 0 pour `--snapshot-retention`.

#### Annotations de planification prises en charge

Le tableau suivant décrit les annotations que vous pouvez utiliser lors de la création d'une demande de changement (CR) de planification :

Annotation	Type	Description	Valeur par défaut
<code>protect.trident.netapp.io/full-backup-rule</code>	chaîne	Spécifie la règle de planification des sauvegardes complètes. Vous pouvez le paramétrier pour <code>always</code> pour une sauvegarde complète et permanente ou personnalisez-la en fonction de vos besoins. Par exemple, si vous choisissez une granularité quotidienne, vous pouvez spécifier les jours de la semaine où la sauvegarde complète doit avoir lieu (par exemple, <code>"Monday, Thursday"</code> ).	Non configuré (toutes les sauvegardes sont incrémentales)
<code>protect.trident.netapp.io/snaps hot-completion-timeout</code>	chaîne	Durée maximale autorisée pour l'exécution de l'opération globale de capture d'instantané.	"60m"
<code>protect.trident.netapp.io/volume-snapshots-ready-to-use-timeout</code>	chaîne	Le délai maximal autorisé pour que les instantanés de volume atteignent l'état prêt à l'emploi.	"30m"
<code>protect.trident.netapp.io/volume-snapshots-created-timeout</code>	chaîne	Durée maximale autorisée pour la création d'instantanés de volume.	"5m"

Annotation	Type	Description	Valeur par défaut
protect.trident.netapp.io/pvc-bind-timeout-sec	chaîne	Délai maximal (en secondes) d'attente pour que les PersistentVolumeClaims (PVC) nouvellement créées atteignent le Bound phase précédent l'échec des opérations.	"1200" (20 minutes)

## 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.

 Assurez-vous que la politique de récupération est définie sur Delete pour supprimer toutes les données de sauvegarde du stockage d'objets. Le paramètre par défaut de la politique est Retain pour éviter toute perte accidentelle de données. Si la politique n'est pas modifiée Delete, les données de sauvegarde resteront dans le stockage d'objets et nécessiteront une suppression manuelle.

### 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.

## Restauration des applications

### 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.
- La restauration à partir d'une sauvegarde vers un espace de noms différent ou vers l'espace de noms d'origine est prise en charge pour les volumes qtree. En revanche, la restauration à partir d'un snapshot vers un espace de noms différent ou vers l'espace de noms d'origine n'est pas prise en charge pour les volumes qtree.
- Vous pouvez utiliser des paramètres avancés pour personnaliser les opérations de restauration. Pour en savoir plus, consultez "[Utiliser les paramètres avancés de restauration de protection Trident](#)".

### 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.

### Avant de commencer

Assurez-vous que l'expiration du jeton de session AWS suffit pour toutes les opérations de restauration s3 à long terme. Si le jeton expire pendant l'opération de restauration, l'opération peut échouer.

- Pour plus d'informations sur la vérification de l'expiration du jeton de session en cours, reportez-vous "[Documentation de l'API AWS](#)" au.
- Pour plus d'informations sur les identifiants avec les ressources AWS, consultez le "[Documentation AWS IAM](#)".

 Lorsque vous restaurez des sauvegardes à l'aide de Kopia comme outil de transfert de données, vous pouvez éventuellement spécifier des annotations dans le CR ou utiliser l'interface de ligne de commande pour contrôler le comportement du stockage temporaire utilisé par Kopia. Se référer à la "[Documentation Kopia](#)" pour plus d'informations sur les options que vous pouvez configurer. Utilisez la `tridentctl-protect create --help` commande pour plus d'informations sur la spécification des annotations avec la CLI Trident Protect.

## 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.

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

3. (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 :



Trident Protect sélectionne automatiquement certaines ressources en raison de leur relation avec les ressources que vous sélectionnez. Par exemple, si vous sélectionnez une ressource de demande de volume persistant et qu'elle dispose d'un pod associé, Trident Protect restaure également le pod associé.

- **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
```

## Utiliser l'interface de ligne de commande

### 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.

### Avant de commencer

Assurez-vous que l'expiration du jeton de session AWS suffit pour toutes les opérations de restauration s3 à long terme. Si le jeton expire pendant l'opération de restauration, l'opération peut échouer.

- Pour plus d'informations sur la vérification de l'expiration du jeton de session en cours, reportez-vous "[Documentation de l'API AWS](#)" au.
- Pour plus d'informations sur les identifiants avec les ressources AWS, consultez le "[Documentation AWS IAM](#)".

 Lorsque vous restaurez des sauvegardes à l'aide de Kopia comme outil de transfert de données, vous pouvez éventuellement spécifier des annotations dans le CR ou utiliser l'interface de ligne de commande pour contrôler le comportement du stockage temporaire utilisé par Kopia. Se référer à la "[Documentation Kopia](#)" pour plus d'informations sur les options que vous pouvez configurer. Utilisez la `tridentctl-protect create --help` commande pour plus d'informations sur la spécification des annotations avec la CLI Trident Protect.

## 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 :



Trident Protect sélectionne automatiquement certaines ressources en raison de leur relation avec les ressources que vous sélectionnez. Par exemple, si vous sélectionnez une ressource de demande de volume persistant et qu'elle dispose d'un pod associé, Trident Protect restaure également le pod associé.

- **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
```

## Utiliser l'interface de ligne de commande

### 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 `<namespace>/<name>`. Par exemple :

```
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.



Lorsque vous restaurez des sauvegardes à l'aide de Kopia comme outil de transfert de données, vous pouvez éventuellement spécifier des annotations dans le CR ou utiliser l'interface de ligne de commande pour contrôler le comportement du stockage temporaire utilisé par Kopia. Se référer à la "[Documentation Kopia](#)" pour plus d'informations sur les options que vous pouvez configurer. Utilisez le `tridentctl-protect create --help` commande pour plus d'informations sur la spécification des annotations avec la CLI Trident Protect.

### 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.
- Assurez-vous que votre environnement local peut se connecter au compartiment de stockage d'objets défini dans l'AppVault CR lors de l'exécution de l'`tridentctl-protect get appvaultcontent` commande. Si les restrictions réseau empêchent l'accès, exécutez plutôt l'interface de ligne de commande Trident Protect à partir d'un pod sur le cluster de destination.
- Assurez-vous que l'expiration du jeton de session AWS suffit pour toutes les opérations de restauration à long terme. Si le jeton expire pendant l'opération de restauration, l'opération peut échouer.
  - Pour plus d'informations sur la vérification de l'expiration du jeton de session en cours, reportez-vous "[Documentation de l'API AWS](#)" au.
  - Pour plus d'informations sur les identifiants avec les ressources AWS, consultez le "[Documentation de l'AWS](#)".

### 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
```

## Utiliser l'interface de ligne de commande

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

### 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.

 SnapshotRestore prend en charge le `spec.storageClassMapping` attribut, mais uniquement lorsque les classes de stockage source et de destination utilisent le même backend de stockage. Si vous tentez de restaurer un `StorageClass` qui utilise un backend de stockage différent, l'opération de restauration échouera.

### Avant de commencer

Assurez-vous que l'expiration du jeton de session AWS suffit pour toutes les opérations de restauration s3 à long terme. Si le jeton expire pendant l'opération de restauration, l'opération peut échouer.

- Pour plus d'informations sur la vérification de l'expiration du jeton de session en cours, reportez-vous "[Documentation de l'API AWS](#)" au.
- Pour plus d'informations sur les identifiants avec les ressources AWS, consultez le "[Documentation AWS IAM](#)".

## 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.

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

3. (*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 :



Trident Protect sélectionne automatiquement certaines ressources en raison de leur relation avec les ressources que vous sélectionnez. Par exemple, si vous sélectionnez une ressource de demande de volume persistant et qu'elle dispose d'un pod associé, Trident Protect restaure également le pod associé.

- **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
```

## Utiliser 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> \
-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.

#### Avant de commencer

Assurez-vous que l'expiration du jeton de session AWS suffit pour toutes les opérations de restauration s3 à long terme. Si le jeton expire pendant l'opération de restauration, l'opération peut échouer.

- Pour plus d'informations sur la vérification de l'expiration du jeton de session en cours, reportez-vous "[Documentation de l'API AWS](#)" au.
- Pour plus d'informations sur les identifiants avec les ressources AWS, consultez le "[Documentation AWS IAM](#)".

## 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 :



Trident Protect sélectionne automatiquement certaines ressources en raison de leur relation avec les ressources que vous sélectionnez. Par exemple, si vous sélectionnez une ressource de demande de volume persistant et qu'elle dispose d'un pod associé, Trident Protect restaure également le pod associé.

- **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
```

## Utiliser l'interface de ligne de commande

### 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}'
```

## Utiliser les paramètres avancés de restauration de protection Trident

Vous pouvez personnaliser les opérations de restauration à l'aide de paramètres avancés tels que les annotations, les paramètres d'espace de noms et les options de stockage pour répondre à vos besoins spécifiques.

### 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 Red Hat OpenShift, il est important de noter le rôle essentiel des annotations d'espace de noms dans les environnements OpenShift. Les annotations d'espace de noms garantissent que les pods restaurés adhèrent aux autorisations et aux configurations de sécurité appropriées définies par les contraintes de contexte de sécurité (SCC) OpenShift et peuvent accéder aux volumes sans problèmes d'autorisation. Pour plus d'informations, reportez-vous à la "[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 :

```
helm upgrade trident-protect --set  
restoreSkipNamespaceAnnotations=<annotation_key_to_skip_1>,<annotation_key  
_to_skip_2> --reuse-values
```

 Lors d'une opération de restauration ou de basculement, toutes les annotations et étiquettes d'espace de noms spécifiées dans `restoreSkipNamespaceAnnotations` et `restoreSkipNamespaceLabels` sont exclues de l'opération de restauration ou de basculement. Assurez-vous que ces paramètres sont configurés lors de l'installation initiale de Helm. Pour en savoir plus, consultez "["Configurer des paramètres supplémentaires du graphique de protection Trident"](#)".

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)	<ul style="list-style-type: none"> <li>annotation.one/key : « updatedvalue »</li> <li>annotation.deux/touche : « vrai »</li> </ul>	<ul style="list-style-type: none"> <li>environnement=production</li> <li>conformité = hipaa</li> <li>name=ns-1</li> </ul>
Espace de noms ns-2 (destination)	<ul style="list-style-type: none"> <li>annotation.un/touche : « vrai »</li> <li>annotation.trois/touche : « false »</li> </ul>	<ul style="list-style-type: none"> <li>role=base de données</li> </ul>

### 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)	<ul style="list-style-type: none"> <li>annotation.one/key : « updatedvalue »</li> <li>annotation.deux/touche : « vrai »</li> <li>annotation.trois/touche : « false »</li> </ul>	<ul style="list-style-type: none"> <li>name=ns-2</li> <li>conformité = hipaa</li> <li>environnement=production</li> <li>role=base de données</li> </ul>

## Champs pris en charge

Cette section décrit les champs supplémentaires disponibles pour les opérations de restauration.

## Mappage des classes de stockage

Le `spec.storageClassMapping` L'attribut définit un mappage d'une classe de stockage présente dans l'application source vers une nouvelle classe de stockage sur le cluster cible. Vous pouvez l'utiliser lors de la

migration d'applications entre des clusters avec différentes classes de stockage ou lors du changement du backend de stockage pour les opérations BackupRestore.

#### Exemple:

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

#### Annotations prises en charge

Cette section répertorie les annotations prises en charge pour configurer différents comportements du système. Si une annotation n'est pas explicitement définie par l'utilisateur, le système utilisera la valeur par défaut.

Annotation	Type	Description	Valeur par défaut
protect.trident.netapp.io/data-mover-timeout-sec	chaîne	Le temps maximal (en secondes) autorisé pour que le fonctionnement du moteur de transfert de données soit bloqué.	"300"
protect.trident.netapp.io/kopia-content-cache-size-limit-mb	chaîne	La limite de taille maximale (en mégaoctets) pour le cache de contenu Kopia.	"1000"
protect.trident.netapp.io/pvc-bind-timeout-sec	chaîne	Délai maximal (en secondes) d'attente pour que les PersistentVolumeClaims (PVC) nouvellement créées atteignent le Bound phase précédant l'échec des opérations. S'applique à tous les types de restauration CR (BackupRestore, BackupInplaceRestore, SnapshotRestore, SnapshotInplaceRestore). Utilisez une valeur plus élevée si votre système de stockage ou votre cluster nécessite souvent plus de temps.	"1200" (20 minutes)

## 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 Red Hat OpenShift, il est important de noter le rôle essentiel des annotations d'espace de noms dans les environnements OpenShift. Les annotations d'espace de noms garantissent que les pods restaurés adhèrent aux autorisations et aux configurations de sécurité appropriées définies par les contraintes de contexte de sécurité (SCC) OpenShift et peuvent accéder aux volumes sans problèmes d'autorisation. Pour plus d'informations, reportez-vous à la "[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 :

```
helm upgrade trident-protect --set
restoreSkipNamespaceAnnotations=<annotation_key_to_skip_1>,<annotation_key
_to_skip_2> --reuse-values
```

 Lors d'une opération de restauration ou de basculement, toutes les annotations et étiquettes d'espace de noms spécifiées dans `restoreSkipNamespaceAnnotations` et `restoreSkipNamespaceLabels` sont exclues de l'opération de restauration ou de basculement. Assurez-vous que ces paramètres sont configurés lors de l'installation initiale de Helm. Pour en savoir plus, consultez "["Configurer des paramètres supplémentaires du graphique de protection Trident"](#)".

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)	<ul style="list-style-type: none"><li>annotation.one/key : « updatedvalue »</li><li>annotation.deux/touche : « vrai »</li></ul>	<ul style="list-style-type: none"><li>environnement=production</li><li>conformité = hipaa</li><li>name=ns-1</li></ul>

Espace de noms	Annotations	Étiquettes
Espace de noms ns-2 (destination)	<ul style="list-style-type: none"> <li>annotation.un/touche : « vrai »</li> <li>annotation.trois/touche : « false »</li> </ul>	<ul style="list-style-type: none"> <li>role=base de données</li> </ul>

## 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)	<ul style="list-style-type: none"> <li>annotation.one/key : « updatedvalue »</li> <li>annotation.deux/touche : « vrai »</li> <li>annotation.trois/touche : « false »</li> </ul>	<ul style="list-style-type: none"> <li>name=ns-2</li> <li>conformité = hipaa</li> <li>environnement=production</li> <li>role=base de données</li> </ul>



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".](#)

## Crochets d'exécution lors des opérations de basculement et d'inversion

Lorsque vous utilisez la relation AppMirror pour protéger votre application, il existe des comportements spécifiques liés aux hooks d'exécution dont vous devez être conscient lors des opérations de basculement et d'inversion.

- Lors du basculement, les hooks d'exécution sont automatiquement copiés du cluster source vers le cluster de destination. Il n'est pas nécessaire de les recréer manuellement. Après le basculement, les hooks d'exécution sont présents dans l'application et s'exécuteront lors de toute action pertinente.
- Lors d'une resynchronisation inverse ou d'une inversion, tous les hooks d'exécution existants sur l'application sont supprimés. Lorsque l'application source devient l'application de destination, ces hooks d'exécution ne sont plus valides et sont supprimés pour empêcher leur exécution.

Pour en savoir plus sur les hooks d'exécution, reportez-vous à ["Gérer les crochets d'exécution de 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. Sur le cluster source, créez un AppVault pour l'application 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> -n  
trident-protect
```

2. Sur le cluster source, créez l'application source CR :

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

- Créez le fichier de ressource personnalisée (CR) et nommez-le (par exemple, `trident-protect-app-source.yaml`).
- 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: {}
```

- 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

- Créez l'application source. Par exemple :

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

- Vous pouvez également, si vous le souhaitez, prendre un instantané de l'application source sur le cluster source. Cet instantané sert de base à l'application sur le cluster de destination. Si vous ignorez cette étape, vous devrez attendre la prochaine capture d'écran planifiée pour obtenir une version récente. Pour créer un instantané à la demande, reportez-vous à "[Créer un snapshot à la demande](#)".

4. Sur le cluster source, créez la ressource personnalisée (CR) de planification de réPLICATION :

Outre le planning ci-dessous, il est recommandé de créer un planning de snapshots quotidien distinct avec une période de conservation de 7 jours afin de conserver un snapshot commun entre les clusters ONTAP appairés. Cela garantit la disponibilité des snapshots pendant 7 jours maximum, mais la période de conservation peut être personnalisée selon les besoins de l'utilisateur.



En cas de basculement, le système peut utiliser ces snapshots pendant sept jours maximum pour les opérations d'inversion. Cette approche rend le processus d'inversion plus rapide et plus efficace, car seules les modifications apportées depuis le dernier snapshot sont transférées, et non l'intégralité des données.

Si un calendrier existant pour l'application répond déjà aux exigences de conservation souhaitées, aucun calendrier supplémentaire n'est requis.

## Créez la planification de réplication à l'aide d'une ressource personnalisée (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** : (*Obligatoire*) Cette valeur doit correspondre au champ `metadata.name` de l'AppVault pour l'application source.
    - **spec.applicationRef**: (*Obligatoire*) Cette valeur doit correspondre au champ `metadata.name` de la ressource personnalisée (CR) de l'application source.
    - **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
  namespace: my-app-namespace
spec:
  appVaultRef: my-appvault-name
  applicationRef: my-app-name
  backupRetention: "0"
  enabled: true
  granularity: Custom
  recurrenceRule: |-  
    DTSTART:20220101T000200Z  
    RRULE:FREQ=MINUTELY;INTERVAL=5
  snapshotRetention: "2"
```

- i. 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
```

#### Créez la planification de réplication à l'aide de l'interface de ligne de commande (CLI).

- Créez la planification de réplication en remplaçant les valeurs entre crochets par les informations provenant de votre environnement :

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

#### Exemple:

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

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

- Appliquer la CR :

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

- Créez une CR AppVault de destination 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 :

- Créez le fichier de ressource personnalisée (CR) et nommez-le (par exemple, `trident-protect-appvault-secondary-destination.yaml`).
- 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 trident-protect
```

8. Sur le cluster de destination, 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.sourceApplicationUID :** (Obligatoire) Cette valeur doit correspondre à l'UID de l'application source que vous avez définie dans la CR de l'application source.
  - **spec.storageClassName:** (*Optionnel*) Choisissez le nom d'une classe de stockage valide sur le cluster. La classe de stockage doit être liée à une machine virtuelle de stockage ONTAP qui est appariée avec l'environnement source. Si la classe de stockage n'est pas spécifiée, la classe de stockage par défaut du cluster sera utilisée.
  - **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-vsimg-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 crochets par les informations provenant de votre environnement :

```
tridentctl-protect create appmirrorrelationship
<name_of_appmirrorrelationship> --destination-app-vault
<my_vault_name> --source-app-vault <my_vault_name> --recurrence
--rule <rule> --namespace-mapping <ns_mapping> --source-app-id
<source_app_UID> --source-app <my_source_app_name> --storage
--class <storage_class_name> -n <application_namespace>
```

**Exemple:**

```
tridentctl-protect create appmirrorrelationship my-amr
--destination-app-vault appvault2 --source-app-vault appvault1
--recurrence-rule
"DTSTART:20220101T000200Z\nRRULE:FREQ=MINUTELY;INTERVAL=5"
--source-app my-app --namespace-mapping "my-source-ns1:my-dest-
ns1,my-source-ns2:my-dest-ns2" --source-app-id 373f24c1-5769-
404c-93c3-5538af6ccc36 --storage-class my-storage-class -n my-
dest-ns1
```

9. (Optional) sur le cluster de destination, 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. Sur le cluster de destination, modifiez 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. Appliquer 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. Facultatif : sur le cluster source, créez un snapshot de l'application source. Cela permet de s'assurer que les dernières modifications du cluster source sont capturées.
2. Sur le cluster de destination, modifiez le fichier CR AppMirrorRelationship (par exemple, `trident-protect-relationship.yaml`) et définissez la valeur `spec.desiredState` sur `Established`.
3. Enregistrez le fichier CR.
4. Appliquer 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. Sur le cluster de destination d'origine, supprimez la CR AppMirrorRelationship. 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 la nouvelle destination (cluster source d'origine) est configurée avec les deux CRS AppVault.
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. Sur le cluster source, créer un snapshot 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. Sur le cluster source, une fois l'instantané d'arrêt terminé, obtenir l'état de l'instantané d'arrêt :

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

3. Sur le cluster source, 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 nouveau cluster de destination vers le nouveau 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 (laissez 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.

## Rétablissement du 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. Sur le cluster de désaturation actuel, supprimez la CR 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 clusters ou vers différentes classes de stockage en restaurant les données de sauvegarde.



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 :
  - "Créer un snapshot".
  - "Créer une sauvegarde".
2. Sur le même cluster, effectuez l'une des opérations suivantes, selon que vous avez créé un snapshot ou une sauvegarde :
  - "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 une sauvegarde vers la classe de stockage de destination.

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` ou `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.



Si un hook d'exécution pré-snapshot ajoute, modifie ou supprime des ressources Kubernetes, ces modifications sont incluses dans l'instantané ou la sauvegarde et dans toute opération de restauration ultérieure.

## 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: IyEvYmluL2Jhc2gKZWNobyAiZXhhbXBsZSBzY3JpcHQiCg==
  timeout: 10
  arguments:
    - FirstExampleArg
    - SecondExampleArg
  matchingCriteria:
    - type: containerName
      value: mysql
    - type: containerImage
      value: bitnami/mysql
    - type: podName
      value: mysql
    - type: namespaceName
      value: mysql-a
    - type: podLabel
      value: app.kubernetes.io/component=primary
    - type: podLabel
      value: helm.sh/chart=mysql-10.1.0
    - type: podLabel
      value: deployment-type=production

```

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

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

## Utiliser l'interface de ligne de commande

### Étapes

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

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

## Exécuter manuellement un crochet d'exécution

Vous pouvez exécuter manuellement un crochet d'exécution à des fins de test ou si vous devez exécuter de nouveau le crochet manuellement après un échec. Vous devez disposer des autorisations propriétaire, administrateur ou membre pour exécuter manuellement les crochets d'exécution.

L'exécution manuelle d'un crochet d'exécution se compose de deux étapes de base :

1. Créez une sauvegarde de ressource, qui collecte les ressources et crée une sauvegarde de celles-ci, en déterminant l'emplacement d'exécution du hook
2. Exécutez le crochet d'exécution contre la sauvegarde

## Étape 1 : création d'une sauvegarde de ressource

## Utiliser une CR

### Étapes

1. Créez le fichier de ressource personnalisée (CR) et nommez-le `trident-protect-resource-backup.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 créer la sauvegarde de ressource.
  - **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}'
```

Exemple YAML :

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

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

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

## Utiliser 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 resourcebackup <my_backup_name> --app  
<my_app_name> --appvault <my_appvault_name> -n  
<my_app_namespace> --app-archive-path <app_archive_path>
```

2. Afficher l'état de la sauvegarde. Vous pouvez utiliser cet exemple de commande plusieurs fois jusqu'à ce que l'opération soit terminée :

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

3. Vérifiez que la sauvegarde a réussi :

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

**Étape 2 : exécutez le crochet d'exécution**

## Utiliser une CR

### Étapes

1. Créez le fichier de ressource personnalisée (CR) et nommez-le `trident-protect-hook-run.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*) Assurez-vous que cette valeur correspond au nom de l'application du CR ResourceBackup que vous avez créé à l'étape 1.
  - **Spec.appVaultRef**: (*required*) Assurez-vous que cette valeur correspond à la référence appVaultRef de la CR ResourceBackup que vous avez créée à l'étape 1.
  - **Spec.appArchivePath** : Assurez-vous que cette valeur correspond à appArchivePath à partir du CR ResourceBackup que vous avez créé à l'étape 1.

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

- **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.stage**: (*required*) Une chaîne indiquant quelle étape de l'action doit être exécutée par le crochet d'exécution. Cette course de crochet ne fera pas courir de crochets dans une autre étape. Valeurs possibles :
  - Pré
  - Post

Exemple YAML :

```

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

```

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

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

## Utiliser l'interface de ligne de commande

### Étapes

1. Créez la demande d'exécution manuelle du crochet :

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

2. Vérifiez l'état de l'exécution du hook. Vous pouvez exécuter cette commande plusieurs fois jusqu'à ce que l'opération soit terminée :

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

3. Décrivez l'objet exehooksrun pour afficher les détails et l'état finaux :

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

# 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 :



Cette étape n'est pas requise pour la version 25.06 et les versions ultérieures.

```
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 JOUSSANCE, 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 DÉLICTUELLE (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.