Utilisez Trident

Trident

NetApp
July 01, 2026

Sommaire

Utilisez Trident

Préparez le nœud de travail

Tous les nœuds de travail du cluster Kubernetes doivent pouvoir monter les volumes que vous avez provisionnés pour vos pods. Pour préparer les nœuds de travail, vous devez installer NFS, iSCSI, NVMe/TCP ou les outils FC en fonction de votre sélection de pilote.

Choisir les bons outils

Si vous utilisez une combinaison de pilotes, vous devez installer tous les outils requis pour vos pilotes. Les versions récentes de Red Hat Enterprise Linux CoreOS (RHCOS) ont les outils installés par défaut.

Outils NFS

"Installez les outils NFS" si vous utilisez : `ontap-nas`, `ontap-nas-economy`, `ontap-nas-flexgroup`, ou `azure-netapp-files`.

Outils iSCSI

"Installez les outils iSCSI" si vous utilisez : `ontap-san`, `ontap-san-economy`, `solidfire-san`.

Outils NVMe

"Installez les outils NVMe" si vous utilisez `ontap-san` pour nonvolatile memory express (NVMe) sur TCP (NVMe/TCP).

> NetApp recommande ONTAP 9.12 ou une version ultérieure pour NVMe/TCP.

Outils SCSI sur FC

Consultez "Méthodes de configuration des hôtes SAN FC et FC-NVMe" pour plus d'informations sur la configuration de vos hôtes SAN FC et FC-NVMe.

"Installez les outils FC" si vous utilisez `ontap-san` avec sanType `fcp` (SCSI sur FC).

Points à prendre en compte : * SCSI sur FC est pris en charge sur OpenShift et KubeVirt environnements. * SCSI sur FC n'est pas pris en charge sur Docker. * L'auto-réparation iSCSI n'est pas applicable à SCSI sur FC.

Outils SMB

"Préparez-vous à provisionner des volumes SMB" si vous utilisez : `ontap-nas` pour provisionner des volumes SMB.

Découverte de service de nœud

Trident tente de détecter automatiquement si le nœud peut exécuter des services iSCSI ou NFS.

> La découverte de services sur les nœuds identifie les services découverts, mais ne garantit pas que les services sont correctement configurés. Inversement, l'absence d'un service découvert ne garantit pas que le montage du volume échouera.

Revoir les événements

Trident crée des événements pour le nœud afin d'identifier les services découverts. Pour consulter ces événements, exécutez :

```
kubectl get event -A --field-selector involvedObject.name=<Kubernetes node name>
```

Examiner les services découverts

Trident identifie les services activés pour chaque nœud sur le Trident node CR. Pour afficher les services détectés, exécutez :

```
tridentctl get node -o wide -n <Trident namespace>
```

Volumes NFS

Installez les outils NFS à l'aide des commandes correspondant à votre système d'exploitation. Assurez-vous que le service NFS est démarré au moment du démarrage.

RHEL 8+

```
sudo yum install -y nfs-utils
```

Ubuntu

```
sudo apt-get install -y nfs-common
```

Redémarrez vos nœuds de travail après l'installation des outils NFS pour éviter les échecs lors de l'attachement des volumes aux conteneurs.

volumes iSCSI

Trident peut établir automatiquement une session iSCSI, analyser les LUN, découvrir les périphériques multipath, les formater et les monter à un pod.

Capacités d'auto-réparation iSCSI

Pour les systèmes ONTAP, Trident exécute une auto-réparation iSCSI toutes les cinq minutes afin de :

1. **Identifier** l'état de session iSCSI souhaité et l'état de session iSCSI actuel.
2. **Comparez** l'état souhaité à l'état actuel pour identifier les réparations nécessaires. Trident détermine les priorités de réparation et le moment où il faut préempter les réparations.
3. **Effectuer les réparations** nécessaires pour ramener l'état actuel de la session iSCSI à l'état souhaité de la session iSCSI.

Les journaux d'activité d'auto-réparation se trouvent dans le `trident-main` conteneur sur le pod Daemonset respectif. Pour afficher les journaux, vous devez avoir défini `debug` sur "true" lors de l'installation de Trident.

Les capacités d'auto-réparation iSCSI de Trident peuvent aider à prévenir :

- Des sessions iSCSI obsolètes ou défaillantes qui pourraient survenir après un problème de connectivité réseau. En cas de session obsolète, Trident attend sept minutes avant de se déconnecter pour rétablir la connexion avec un portail.

  Par exemple, si les secrets CHAP sont renouvelés sur le contrôleur de stockage et que le réseau perd la connectivité, les anciens secrets CHAP (obsolètes) peuvent persister. L'auto-réparation peut reconnaître cela et rétablir automatiquement la session pour appliquer les secrets CHAP mis à jour.

- Sessions iSCSI manquantes
- LUN manquantes

Points à prendre en compte avant de mettre à niveau Trident

- Si seuls les igroups par nœud (introduits dans 23.04+) sont utilisés, l'auto-réparation iSCSI lancera des analyses SCSI pour tous les périphériques du bus SCSI.
- Si seuls les igroups à portée backend (dépréciés depuis 23.04) sont utilisés, l'auto-réparation iSCSI lancera des analyses SCSI pour les ID LUN exacts sur le bus SCSI.
- Si une combinaison d'igroups par nœud et d'igroups à portée backend est utilisée, l'auto-réparation iSCSI lancera des analyses SCSI pour les ID LUN exacts sur le bus SCSI.

Installez les outils iSCSI

Installez les outils iSCSI en utilisant les commandes pour votre système d'exploitation.

Avant de commencer

- Chaque nœud du cluster Kubernetes doit posséder un IQN unique. **Il s'agit d'une condition préalable indispensable**.
- Si vous utilisez RHCOS version 4.5 ou ultérieure, ou une autre distribution Linux compatible RHEL, avec le `solidfire-san` pilote et Element OS 12.5 ou une version antérieure, assurez-vous que l'algorithme d'authentification CHAP est défini sur MD5 dans `/etc/iscsi/iscsid.conf`. Les algorithmes CHAP sécurisés conformes à la norme FIPS, SHA1, SHA-256 et SHA3-256 sont disponibles avec Element 12.7.

```
sudo sed -i 's/^\(node.session.auth.chap_algs\).*/\1 = MD5/'
/etc/iscsi/iscsid.conf
```

- Lors de l'utilisation de nœuds de travail exécutant RHEL/Red Hat Enterprise Linux CoreOS (RHCOS) avec des volumes persistants iSCSI, spécifiez la `discard` mountOption dans la StorageClass pour effectuer la récupération d'espace en ligne. Consultez "Documentation Red Hat".
- Assurez-vous d'avoir effectué la mise à jour vers la dernière version du `multipath-tools`.

RHEL 8+

1. Installez les paquets système suivants :

```
sudo yum install -y lsscsi iscsi-initiator-utils device-mapper-
multipath
```

2. Vérifiez que la version de iscsi-initiator-utils est 6.2.0.874-2.el7 ou ultérieure :

```
rpm -q iscsi-initiator-utils
```

3. Définissez la numérisation sur manuel :

```
sudo sed -i 's/^\(node.session.scan\).*/\1 = manual/'
/etc/iscsi/iscsid.conf
```

4. Activez le multipathing :

```
sudo mpathconf --enable --with_multipathd y --find_multipaths n
```

> Assurez-vous `/etc/multipath.conf` contient `find_multipaths no` sous `defaults`.

5. Assurez-vous que `iscsid` et `multipathd` sont en cours d'exécution :

```
sudo systemctl enable --now iscsid multipathd
```

6. Activer et démarrer `iscsi`:

```
sudo systemctl enable --now iscsi
```

Ubuntu

1. Installez les paquets système suivants :

```
sudo apt-get install -y open-iscsi lsscsi sg3-utils multipath-tools
scsitools
```

2. Vérifiez que la version d'open-iscsi est 2.0.874-5ubuntu2.10 ou ultérieure (pour bionic) ou 2.0.874-7.1ubuntu6.1 ou ultérieure (pour focal):

```
dpkg -l open-iscsi
```

3. Définissez la numérisation sur manuel :

```
sudo sed -i 's/^\(node.session.scan\).*/\1 = manual/'
/etc/iscsi/iscsid.conf
```

4. Activez le multipathing :

```
sudo tee /etc/multipath.conf <<-EOF
defaults {
    user_friendly_names yes
    find_multipaths no
}
EOF
sudo systemctl enable --now multipath-tools.service
sudo service multipath-tools restart
```

> Assurez-vous `/etc/multipath.conf` contient `find_multipaths no` sous `defaults`.

5. Assurez-vous que `open-iscsi` et `multipath-tools` sont activés et en cours d'exécution :

```
sudo systemctl status multipath-tools
sudo systemctl enable --now open-iscsi.service
sudo systemctl status open-iscsi
```

> Pour Ubuntu 18.04, vous devez découvrir les ports cibles avec `iscsiadm` avant de démarrer `open-iscsi` pour que le démon iSCSI démarre. Vous pouvez également modifier le service `iscsi` pour qu'il démarre `iscsid` automatiquement.

Configurer ou désactiver l'autoréparation iSCSI

Vous pouvez configurer les paramètres d'auto-réparation iSCSI Trident suivants pour corriger les sessions obsolètes :

- **Intervalle d'auto-réparation iSCSI** : détermine la fréquence à laquelle l'auto-réparation iSCSI est invoquée (par défaut : 5 minutes). Vous pouvez le configurer pour qu'il s'exécute plus fréquemment en définissant une valeur plus petite ou moins fréquemment en définissant une valeur plus grande.

Définir l'intervalle d'auto-réparation iSCSI à 0 désactive complètement l'auto-réparation iSCSI. Nous déconseillons de désactiver l'auto-réparation iSCSI ; elle ne doit être désactivée que dans certains scénarios lorsque l'auto-réparation iSCSI ne fonctionne pas comme prévu ou à des fins de débogage.

- **Délai d'attente d'auto-réparation iSCSI** : détermine la durée pendant laquelle l'auto-réparation iSCSI attend avant de se déconnecter d'une session défaillante et de tenter de se reconnecter (par défaut : 7 minutes). Vous pouvez le configurer sur une valeur plus élevée afin que les sessions identifiées comme défaillantes doivent attendre plus longtemps avant d'être déconnectées puis qu'une tentative de reconnexion soit effectuée, ou sur une valeur plus faible pour se déconnecter et se reconnecter plus tôt.

Helm

Pour configurer ou modifier les paramètres d'auto-réparation iSCSI, transmettez les `iscsiSelfHealingInterval` et `iscsiSelfHealingWaitTime` paramètres lors de l'installation ou de la mise à jour de helm.

L'exemple suivant configure l'intervalle d'auto-réparation iSCSI à 3 minutes et le délai d'attente d'auto-réparation à 6 minutes :

```
helm install trident trident-operator-100.2602.0.tgz --set
iscsiSelfHealingInterval=3m0s --set iscsiSelfHealingWaitTime=6m0s -n
trident
```

tridentctl

Pour configurer ou modifier les paramètres d'auto-réparation iSCSI, transmettez les `iscsi-self-healing-interval` et `iscsi-self-healing-wait-time` paramètres lors de l'installation ou de la mise à jour de tridentctl.

L'exemple suivant configure l'intervalle d'auto-réparation iSCSI à 3 minutes et le délai d'attente d'auto-réparation à 6 minutes :

```
tridentctl install --iscsi-self-healing-interval=3m0s --iscsi-self
-healing-wait-time=6m0s -n trident
```

Volumes NVMe/TCP

Installez les outils NVMe à l'aide des commandes pour votre système d'exploitation.

- NVMe nécessite RHEL 9 ou version ultérieure.
- Si la version du noyau de votre nœud Kubernetes est trop ancienne ou si le package NVMe n'est pas disponible pour votre version du noyau, vous devrez peut-être mettre à jour la version du noyau de votre nœud vers une version avec le package NVMe.

RHEL 9

```
sudo yum install nvme-cli
sudo yum install linux-modules-extra-$(uname -r)
sudo modprobe nvme-tcp
```

Ubuntu

```
sudo apt install nvme-cli
sudo apt -y install linux-modules-extra-$(uname -r)
sudo modprobe nvme-tcp
```

Vérifier l'installation

Après l'installation, vérifiez que chaque nœud du cluster Kubernetes possède un NQN unique à l'aide de la commande :

```
cat /etc/nvme/hostnqn
```

> Trident modifie la valeur `ctrl_device_tmo` pour garantir que NVMe ne renonce pas au chemin s'il tombe en panne. Ne modifiez pas ce paramètre.

Volumes SCSI sur FC

Vous pouvez désormais utiliser le protocole Fibre Channel (FC) avec Trident pour provisionner et gérer les ressources de stockage sur le système ONTAP.

Prérequis

Configurez les paramètres réseau et de nœud requis pour FC.

Paramètres réseau

1. Obtenez le WWPN des interfaces cibles. Consultez "network interface show" pour plus d'informations.
2. Obtenez le WWPN pour les interfaces sur l'initiateur (Host).

   Consultez les utilitaires correspondants du système d'exploitation hôte.

3. Configurez le zonage sur le commutateur FC en utilisant les WWPN du Host et du target.

   Reportez-vous à la documentation du fournisseur du commutateur concerné pour obtenir des informations.

   Pour plus de détails, veuillez consulter la documentation ONTAP suivante :

   - "Aperçu du zonage Fibre Channel et FCoE"
   - "Méthodes de configuration des hôtes SAN FC et FC-NVMe"

Installez les outils FC

Installez les outils FC à l'aide des commandes correspondant à votre système d'exploitation.

- Lors de l'utilisation de nœuds de travail exécutant RHEL/Red Hat Enterprise Linux CoreOS (RHCOS) avec des FC PVs, spécifiez le `discard` mountOption dans le StorageClass pour effectuer une récupération d'espace en ligne. Consultez "Documentation Red Hat".

RHEL 8+

1. Installez les paquets système suivants :

```
sudo yum install -y lsscsi device-mapper-multipath
```

2. Activez le multipathing :

```
sudo mpathconf --enable --with_multipathd y --find_multipaths n
```

> Assurez-vous `/etc/multipath.conf` contient `find_multipaths no` sous `defaults`.

3. Assurez-vous que `multipathd` est en cours d'exécution :

```
sudo systemctl enable --now multipathd
```

Ubuntu

1. Installez les paquets système suivants :

```
sudo apt-get install -y lsscsi sg3-utils multipath-tools scsitools
```

2. Activez le multipathing :

```
sudo tee /etc/multipath.conf <<-EOF
defaults {
    user_friendly_names yes
    find_multipaths no
}
EOF
sudo systemctl enable --now multipath-tools.service
sudo service multipath-tools restart
```

> Assurez-vous `/etc/multipath.conf` contient `find_multipaths no` sous `defaults`.

3. Assurez-vous que `multipath-tools` est activé et en cours d'exécution :

```
sudo systemctl status multipath-tools
```

Préparez-vous à provisionner des volumes SMB

Vous pouvez provisionner des volumes SMB à l'aide de `ontap-nas`drivers.

> Vous devez configurer les protocoles NFS et SMB/CIFS sur la SVM pour créer un `ontap-nas-economy` volume SMB pour les clusters ONTAP sur site. L'absence de configuration de l'un ou l'autre de ces protocoles entraînera l'échec de la création du volume SMB.

> `autoExportPolicy` n'est pas pris en charge pour les volumes SMB.

Avant de commencer

Avant de pouvoir provisionner des volumes SMB, vous devez disposer des éléments suivants.

- Un cluster Kubernetes avec un nœud contrôleur Linux et au moins un nœud de travail Windows exécutant Windows Server 2022. Trident prend uniquement en charge les volumes SMB montés sur des pods exécutés sur des nœuds Windows.
- Au moins un secret Trident contenant vos identifiants Active Directory. Pour générer le secret `smbcreds` :

```
kubectl create secret generic smbcreds --from-literal username=user
--from-literal password='password'
```

- Un proxy CSI configuré en tant que service Windows. Pour configurer un `csi-proxy`, reportez-vous à "GitHub : CSI Proxy" ou "GitHub: CSI Proxy pour Windows" pour les nœuds Kubernetes exécutés sous Windows.

Étapes

1. Pour ONTAP sur site, vous pouvez éventuellement créer un partage SMB ou Trident peut en créer un pour vous.

   > Les partages SMB sont requis pour Amazon FSx pour ONTAP.

   Vous pouvez créer les partages d'administration SMB de deux manières : soit à l'aide du "Microsoft Management Console" Shared Folders snap-in, soit à l'aide de l'interface de ligne de commande ONTAP. Pour créer les partages SMB à l'aide de l'interface de ligne de commande ONTAP :

   a. Si nécessaire, créez la structure du chemin d'accès du répertoire pour le partage.

      La `vserver cifs share create`commande vérifie le chemin spécifié dans l'option -path lors de la création du partage. Si le chemin spécifié n'existe pas, la commande échoue.

   b. Créer un partage SMB associé à la SVM spécifiée :

```
vserver cifs share create -vserver vserver_name -share-name
share_name -path path [-share-properties share_properties,...]
[other_attributes] [-comment text]
```

   c. Vérifiez que le partage a été créé :

```
vserver cifs share show -share-name share_name
```

Reportez-vous à "Créer un partage SMB" pour plus de détails.

2. Lors de la création du backend, vous devez configurer les éléments suivants pour spécifier les volumes SMB. Pour toutes les options de configuration du backend FSx pour ONTAP, reportez-vous à "Options de configuration et exemples pour FSx for ONTAP".

Paramètre	Description	Exemple
`smbShare`	Vous pouvez spécifier l'une des options suivantes : le nom d'un partage SMB créé à l'aide de la console de gestion Microsoft ou de l'interface de ligne de commande ONTAP ; un nom pour permettre à Trident de créer le partage SMB ; ou vous pouvez laisser le paramètre vide pour empêcher l'accès partagé aux volumes. Ce paramètre est facultatif pour ONTAP sur site. Ce paramètre est obligatoire pour Amazon FSx for ONTAP backends et ne peut pas être vide.	`smb-share`
`nasType`	**Doit être défini sur `smb`.** Si nul, la valeur par défaut est `nfs`.	`smb`
`securityStyle`	Style de sécurité pour les nouveaux volumes. **Doit être défini sur `ntfs` ou `mixed` pour les volumes SMB.**	`ntfs` or `mixed` pour les volumes SMB
`unixPermissions`	Mode pour les nouveaux volumes. **Doit rester vide pour les volumes SMB.**	""

Configurer et gérer les backends

Configurer les backends

Un backend définit la relation entre Trident et un système de stockage. Il indique à Trident comment communiquer avec ce système de stockage et comment Trident doit provisionner des volumes à partir de celui-ci.

Trident propose automatiquement des pools de stockage provenant de backends correspondant aux exigences définies par une classe de stockage. Découvrez comment configurer le backend pour votre système de stockage.

- "Configurer un backend Azure NetApp Files"
- "Configurer un backend Google Cloud NetApp Volumes"
- "Configurer un backend NetApp HCI ou SolidFire"
- "Configurez un backend avec les pilotes NAS ONTAP ou Cloud Volumes ONTAP"
- "Configurez un backend avec les pilotes SAN ONTAP ou Cloud Volumes ONTAP"

- "Utilisez Trident avec Amazon FSx for NetApp ONTAP"

Azure NetApp Files

Configurer un backend Azure NetApp Files

Utilisez Azure NetApp Files comme backend pour Trident. Ce backend prend en charge les volumes NFS et SMB. Trident prend en charge les identités managées et l'identité de charge de travail pour les clusters Azure Kubernetes Service (AKS).

Environnements cloud Azure pris en charge

Trident prend en charge les backends Azure NetApp Files dans plusieurs environnements cloud Azure.

Les clouds Azure pris en charge incluent :

- Azure Commercial
- Azure Government (Azure Government / MAG)

Lorsque vous déployez Trident ou configurez un backend Azure NetApp Files, assurez-vous qu'Azure Resource Manager et les points de terminaison d'authentification correspondent à votre environnement cloud.

Vérifiez la prise en charge du pilote Azure NetApp Files

Trident fournit le pilote de stockage Azure NetApp Files suivant.

Les modes d'accès pris en charge incluent *ReadWriteOnce* (RWO), *ReadOnlyMany* (ROX), *ReadWriteMany* (RWX) et *ReadWriteOncePod* (RWOP).

Pilote	Protocole	volumeMode	Modes d'accès pris en charge	Systèmes de fichiers pris en charge
`azure-netapp-files`	NFS SMB	Système de fichiers	RWO, ROX, RWX, RWOP	`nfs`, `smb`

Considérations à examiner

- Azure NetApp Files ne prend pas en charge les volumes inférieurs à 50 Gio. Trident crée un volume de 50 Gio lorsqu'un volume plus petit est demandé.
- Trident prend uniquement en charge les volumes SMB montés sur des pods exécutés sur des nœuds Windows.
- Les déploiements Azure NetApp Files dans les clouds Azure non commerciaux nécessitent des points de terminaison Azure Resource Manager et d'authentification spécifiques au cloud. Assurez-vous que Trident et toute configuration backend utilisent les points de terminaison appropriés pour votre environnement cloud Azure.

Utilisez des identités gérées pour AKS

Trident prend en charge "identités gérées" pour les clusters AKS.

Si vous utilisez `tridentctl` pour créer ou gérer des backends Azure NetApp Files, assurez-vous qu'il est configuré pour le bon environnement cloud Azure.

Pour utiliser les identités gérées, vous devez avoir :

- Un cluster Kubernetes déployé à l'aide d'AKS
- Identités gérées configurées sur le cluster Kubernetes AKS
- Trident installé avec `cloudProvider` défini sur `"Azure"`

Opérateur Trident

Modifier `tridentorchestrator_cr.yaml` et définir `cloudProvider` sur `"Azure"`.

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  imagePullPolicy: IfNotPresent
  cloudProvider: "Azure"
```

Helm

L'exemple suivant installe Trident et configure `cloudProvider` en utilisant la variable d'environnement `$CP` :

```
helm install trident trident-operator-100.2602.0.tgz --create-namespace
--namespace <trident-namespace> --set cloudProvider=$CP
```

<code>tridentctl</code>

L'exemple suivant installe Trident et définit le `cloud-provider` indicateur sur `Azure` :

```
tridentctl install --cloud-provider="Azure" -n trident
```

Utiliser l'identité de charge de travail pour AKS

L'identité de charge de travail permet aux pods Kubernetes d'accéder aux ressources Azure en s'authentifiant en tant qu'identité de charge de travail.

Si vous utilisez `tridentctl` pour créer ou gérer des backends Azure NetApp Files, assurez-vous qu'il est configuré pour le bon environnement cloud Azure.

Pour utiliser l'identité de charge de travail, vous devez avoir :

- Un cluster Kubernetes déployé à l'aide d'AKS
- Identité de la charge de travail et oidc-issuer configurés sur le cluster Kubernetes AKS

- Trident installé avec `cloudProvider` défini sur `"Azure"` et `cloudIdentity` défini sur la valeur d'identité de la charge de travail

Opérateur Trident

Modifier `tridentorchestrator_cr.yaml` et définir `cloudProvider` sur `"Azure"`. Définir `cloudIdentity` sur `azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxx`.

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  imagePullPolicy: IfNotPresent
  cloudProvider: "Azure"
  cloudIdentity: 'azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxx' # Edit
```

Helm

Définissez les valeurs des indicateurs **cloud-provider (CP)** et **cloud-identity (CI)** à l'aide des variables d'environnement suivantes :

```
export CP="Azure"
export CI="'azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxx'"
```

L'exemple suivant installe Trident et configure `cloudProvider` en utilisant `$CP` et configure `cloudIdentity` en utilisant `$CI` :

```
helm install trident trident-operator-100.6.0.tgz --set cloudProvider=$CP --set cloudIdentity="$CI"
```

<code>tridentctl</code>

Définissez les valeurs pour les indicateurs **fournisseur de cloud** et **identité cloud** à l'aide des variables d'environnement suivantes :

```
export CP="Azure"
export CI="azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxx"
```

L'exemple suivant installe Trident et définit `cloud-provider` sur `$CP` et `cloud-identity` sur `$CI` :

```
tridentctl install --cloud-provider=$CP --cloud-identity="$CI" -n trident
```

Préparez-vous à configurer un backend Azure NetApp Files

Avant de pouvoir configurer votre backend Azure NetApp Files, vous devez vous assurer que les conditions suivantes sont remplies.

Environnements cloud Azure pris en charge

Trident prend en charge les backends Azure NetApp Files dans plusieurs environnements cloud Azure.

Les clouds Azure pris en charge incluent :

- Azure Commercial
- Azure Government (Azure Government / MAG)

Lors de la préparation de votre environnement, assurez-vous que votre abonnement Azure, votre configuration d'identité et vos ressources Azure NetApp Files sont créés dans l'environnement cloud Azure approprié.

Prérequis pour les volumes NFS et SMB

Si vous utilisez Azure NetApp Files pour la première fois ou dans un nouvel emplacement, une configuration initiale est nécessaire pour configurer Azure NetApp Files et créer un volume NFS. Consultez "Azure : Configurer Azure NetApp Files et créer un volume NFS".

Pour configurer et utiliser un "Azure NetApp Files" backend, vous avez besoin des éléments suivants :

- `subscriptionID`, `tenantID`, `clientID`, `location` et `clientSecret` sont facultatifs lors de l'utilisation d'identités gérées sur un cluster AKS.
- `tenantID`, `clientID`, et `clientSecret` sont facultatifs lors de l'utilisation d'une identité cloud sur un cluster AKS.
- Les déploiements Azure NetApp Files dans les clouds Azure non commerciaux nécessitent des points de terminaison Azure Resource Manager et d'authentification spécifiques au cloud. Assurez-vous que Trident et toute configuration backend utilisent les points de terminaison appropriés pour votre environnement cloud Azure.

- Un pool de capacité. Consultez "Microsoft : Créer un pool de capacité pour Azure NetApp Files".
- Un sous-réseau délégué à Azure NetApp Files. Consultez "Microsoft : Déléguer un sous-réseau à Azure NetApp Files".
- `subscriptionID` à partir d'un abonnement Azure avec Azure NetApp Files activé.
- `tenantID`, `clientID`, et `clientSecret` à partir d'un "Inscription de l'application" dans Azure Active Directory disposant des autorisations suffisantes pour le service Azure NetApp Files. L'inscription de l'application doit utiliser l'une des méthodes suivantes :
  - Le rôle de propriétaire ou de contributeur "prédéfini par Azure".
  - Un "Rôle Contributor personnalisé" au niveau de l'abonnement (`assignableScopes`) avec les autorisations suivantes, limitées uniquement à ce dont Trident a besoin. Après avoir créé le rôle personnalisé, "Attribuez le rôle à l'aide du portail Azure".

Rôle de contributeur personnalisé

```
{
  "id": "/subscriptions/<subscription-
id>/providers/Microsoft.Authorization/roleDefinitions/<role-
definition-id>",
  "properties": {
    "roleName": "custom-role-with-limited-perms",
    "description": "custom role providing limited permissions",
    "assignableScopes": [
      "/subscriptions/<subscription-id>"
    ],
    "permissions": [
      {
        "actions": [
          "Microsoft.NetApp/netAppAccounts/capacityPools/read",
          "Microsoft.NetApp/netAppAccounts/capacityPools/write",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/read",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/write",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/delete",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/snapshots/
read",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/snapshots/
write",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/snapshots/
delete",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/MountTarge
ts/read",
          "Microsoft.Network/virtualNetworks/read",
          "Microsoft.Network/virtualNetworks/subnets/read",

"Microsoft.Features/featureProviders/subscriptionFeatureRegistrat
ions/read",

"Microsoft.Features/featureProviders/subscriptionFeatureRegistrat
ions/write",

"Microsoft.Features/featureProviders/subscriptionFeatureRegistrat
ions/delete",
```

```
            "Microsoft.Features/features/read",
            "Microsoft.Features/operations/read",
            "Microsoft.Features/providers/features/read",

"Microsoft.Features/providers/features/register/action",

"Microsoft.Features/providers/features/unregister/action",

"Microsoft.Features/subscriptionFeatureRegistrations/read"
          ],
          "notActions": [],
          "dataActions": [],
          "notDataActions": []
        }
      ]
    }
}
```

- L'Azure `location` qui contient au moins un "sous-réseau délégué". À partir de Trident 22.01, le paramètre `location` est un champ obligatoire au niveau supérieur du fichier de configuration du backend. Les valeurs d'emplacement spécifiées dans les pools virtuels sont ignorées.
- Pour utiliser `Cloud Identity`, obtenez l' `client ID` à partir d'un "identité gérée attribuée par l'utilisateur" et spécifiez cet ID dans `azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxx`.

Exigences supplémentaires pour les volumes SMB

Pour créer un volume SMB, vous devez disposer de :

- Active Directory configuré et connecté à Azure NetApp Files. Consultez "Microsoft : Créer et gérer des connexions Active Directory pour Azure NetApp Files".
- Un cluster Kubernetes avec un nœud contrôleur Linux et au moins un nœud de travail Windows exécutant Windows Server 2022. Trident prend uniquement en charge les volumes SMB montés sur des pods exécutés sur des nœuds Windows.
- Au moins un secret Trident contenant vos identifiants Active Directory afin qu'Azure NetApp Files puisse s'authentifier auprès d'Active Directory. Pour générer le secret `smbcreds` :

```
kubectl create secret generic smbcreds --from-literal username=user
--from-literal password='password'
```

- Un proxy CSI configuré en tant que service Windows. Pour configurer un `csi-proxy`, reportez-vous à "GitHub : CSI Proxy" ou "GitHub: CSI Proxy pour Windows" pour les nœuds Kubernetes exécutés sous Windows.

Options et exemples de configuration du backend Azure NetApp Files

Découvrez les options de configuration NFS et SMB pour Azure NetApp Files et consultez des exemples de configuration.

Options de configuration du backend

Trident utilise votre configuration backend (sous-réseau, réseau virtuel, niveau de service et emplacement) pour créer des volumes Azure NetApp Files sur des pools de capacité disponibles dans l'emplacement demandé et correspondant au niveau de service et au sous-réseau demandés.

Les backends Azure NetApp Files offrent ces options de configuration.

Paramètre	Description	Défaut
`version`	Version de la configuration du backend.	Toujours 1
`storageDriverName`	Nom du pilote de stockage	"azure-netapp-files"
`backendName`	Nom personnalisé pour le backend de stockage	Driver name + "_" + caractères aléatoires
`subscriptionID`	L'ID d'abonnement de votre abonnement Azure Facultatif lorsque les identités gérées sont activées sur un cluster AKS.	
`tenantID`	L'identifiant du locataire issu d'un enregistrement d'application. Facultatif lorsque des identités gérées ou une identité cloud sont utilisées sur un cluster AKS.	
`clientID`	L'identifiant client issu d'un enregistrement d'application, facultatif lorsque des identités gérées ou une identité cloud sont utilisées sur un cluster AKS.	
`clientSecret`	Le secret client issu d'un App Registration est facultatif lorsque des identités gérées ou une cloud identity sont utilisées sur un cluster AKS.	
`serviceLevel`	L'un de `Standard`, `Premium` ou `Ultra`	"" (aléatoire)
`location`	Nom de l'emplacement Azure où les nouveaux volumes seront créés Facultatif lorsque les identités gérées sont activées sur un cluster AKS.	
`resourceGroups`	Liste des groupes de ressources pour filtrer les ressources découvertes	"[]" (aucun filtre)

Paramètre	Description	Défaut
`netappAccounts`	Liste des comptes NetApp pour le filtrage des ressources découvertes	"[]" (aucun filtre)
`capacityPools`	Liste des pools de capacité pour le filtrage des ressources découvertes	"[]" (no filter, aléatoire)
`virtualNetwork`	Nom d'un réseau virtuel avec un sous-réseau délégué	""
`subnet`	Nom d'un sous-réseau délégué à `Microsoft.Netapp/volumes`	""
`networkFeatures`	Ensemble de fonctionnalités VNet pour un volume, peut être `Basic` ou `Standard`. Les fonctionnalités réseau ne sont pas disponibles dans toutes les régions et peuvent devoir être activées dans un abonnement. Spécifier `networkFeatures` lorsque la fonctionnalité n'est pas activée entraîne l'échec du provisionnement du volume.	""
`nfsMountOptions`	Contrôle précis des options de montage NFS. Ignoré pour les volumes SMB. Pour monter des volumes à l'aide de NFS version 4.1, incluez `nfsvers=4` dans la liste des options de montage séparées par des virgules pour choisir NFS v4.1. Les options de montage définies dans une définition de classe de stockage remplacent celles définies dans la configuration du backend.	"nfsvers=3"
`limitVolumeSize`	Échec de l'approvisionnement si la taille du volume demandée est supérieure à cette valeur	"" (non appliqué par défaut)
`debugTraceFlags`	Options de débogage à utiliser lors du dépannage. Exemple, `\{"api": false, "method": true, "discovery": true}`. N'utilisez cette option que si vous effectuez un dépannage et avez besoin d'un journal détaillé.	null
`nasType`	Configurez la création de volumes NFS ou SMB. Les options sont `nfs`, `smb` ou null. La valeur null correspond par défaut à des volumes NFS.	`nfs`

Paramètre	Description	Défaut
`supportedTopologies`	Représente une liste de régions et de zones prises en charge par ce backend. Pour plus d'informations, consultez "Utiliser la topologie CSI".	
`qosType`	Représente le type de QoS : Auto ou Manual.	Automatique
`maxThroughput`	Définit le débit maximal autorisé en Mio/s. Pris en charge uniquement pour les pools de capacité QoS manuels.	`4 MiB/sec`

Pour plus d'informations sur les fonctionnalités réseau, veuillez consulter "Configurer les fonctionnalités réseau pour un volume Azure NetApp Files".

Envisagez les environnements cloud Azure (26.02)

À partir de la version 26.02, Trident prend en charge la création et la gestion de backends Azure NetApp Files dans plusieurs environnements cloud Azure.

Les clouds Azure pris en charge incluent :

- Azure Commercial
- Azure Government (Azure Government / MAG)

Lorsque vous déployez Trident ou créez un backend Azure NetApp Files, assurez-vous que Azure Resource Manager et les points de terminaison d'authentification correspondent à votre environnement cloud. Si les points de terminaison ne correspondent pas, `tridentctl` ne peut pas s'authentifier et la création du backend échoue.

Autorisations et ressources requises

Si vous recevez l'erreur « Aucun pool de capacité trouvé » lors de la création d'un PVC, il est probable que l'enregistrement de votre application ne dispose pas des autorisations et ressources requises (sous-réseau, réseau virtuel, pool de capacité) associées. Si le débogage est activé, Trident enregistre les ressources Azure détectées lors de la création du backend. Vérifiez qu'un rôle approprié est utilisé.

Les valeurs pour `resourceGroups`, `netappAccounts`, `capacityPools`, `virtualNetwork` et `subnet` peuvent être spécifiées à l'aide de noms courts ou de noms complets. Les noms complets sont recommandés dans la plupart des situations, car les noms courts peuvent correspondre à plusieurs ressources portant le même nom.

Si le vNet est situé dans un groupe de ressources différent de celui du compte de stockage Azure NetApp Files (ANF), spécifiez le groupe de ressources du réseau virtuel lors de la configuration de la liste resourceGroups pour le backend.

Les `resourceGroups`, `netappAccounts`, et `capacityPools` valeurs sont des filtres qui limitent l'ensemble des ressources découvertes à celles disponibles pour ce système de stockage et peuvent être spécifiées dans n'importe quelle combinaison. Les noms pleinement qualifiés suivent ce format :

Type	Format
Groupe de ressources	<resource group>
Compte NetApp	<resource group>/<netapp account>
Pool de capacité	<resource group>/<netapp account>/<capacity pool>
Réseau virtuel	<resource group>/<virtual network>
Sous-réseau	<resource group>/<virtual network>/<subnet>

Provisionnement de volumes

Vous pouvez contrôler le provisionnement des volumes par défaut en spécifiant les options suivantes dans une section spéciale du fichier de configuration. Consultez Exemples de configurations pour plus de détails.

Paramètre	Description	Défaut
`exportRule`	Règles d'exportation pour les nouveaux volumes. `exportRule` doit être une liste séparée par des virgules de toute combinaison d'adresses IPv4 ou de sous-réseaux IPv4 en notation CIDR. Ignoré pour les volumes SMB.	"0.0.0.0/0"
`snapshotDir`	Accès au `.snapshot` répertoire	`true`, `false` (Défini explicitement).
`size`	La taille par défaut des nouveaux volumes	"100G"
`unixPermissions`	Les permissions Unix des nouveaux volumes (4 chiffres octaux). Ignoré pour les volumes SMB.	"" (fonctionnalité en avant-première, nécessite une inscription sur la liste blanche dans l'abonnement)

Exemples de configurations

Les exemples suivants présentent des configurations de base qui laissent la plupart des paramètres par défaut. C'est la manière la plus simple de définir un backend.

Configuration minimale

Il s'agit de la configuration minimale absolue du backend. Avec cette configuration, Trident découvre tous vos comptes NetApp, pools de capacité et sous-réseaux délégués à Azure NetApp Files dans l'emplacement configuré, et place les nouveaux volumes aléatoirement sur l'un de ces pools et sous-réseaux. Parce que `nasType` est omis, la `nfs` configuration par défaut s'applique et le backend provisionnera des volumes NFS.

Cette configuration est idéale lorsque vous débutez avec Azure NetApp Files et que vous faites des essais, mais en pratique vous souhaiterez définir une portée supplémentaire pour les volumes que vous provisionnez.

```
---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-anf-1
  namespace: trident
spec:
  version: 1
  storageDriverName: azure-netapp-files
  subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
  tenantID: 68e4f836-edc1-fake-bff9-b2d865ee56cf
  clientID: dd043f63-bf8e-fake-8076-8de91e5713aa
  clientSecret: SECRET
  location: eastus
```

Identités managées pour AKS

Cette configuration backend omet `subscriptionID`, `tenantID`, `clientID` et `clientSecret`, qui sont optionnels lors de l'utilisation d'identités gérées.

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-anf-1
  namespace: trident
spec:
  version: 1
  storageDriverName: azure-netapp-files
  capacityPools:
    - resource-group-1/netapp-account-1/ultra-pool
  resourceGroups:
    - resource-group-1
  netappAccounts:
    - resource-group-1/netapp-account-1
  virtualNetwork: resource-group-1/eastus-prod-vnet
  subnet: resource-group-1/eastus-prod-vnet/eastus-anf-subnet
```

Identité cloud pour AKS

Cette configuration backend omet `tenantID`, `clientID` et `clientSecret`, qui sont optionnels lors de l'utilisation d'une identité cloud.

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-anf-1
  namespace: trident
spec:
  version: 1
  storageDriverName: azure-netapp-files
  capacityPools:
    - ultra-pool
  resourceGroups:
    - aks-ami-eastus-rg
  netappAccounts:
    - smb-na
  virtualNetwork: eastus-prod-vnet
  subnet: eastus-anf-subnet
  location: eastus
  subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
```

Configuration spécifique du niveau de service avec filtres de pool de capacité

Cette configuration backend place les volumes dans l'emplacement `eastus` d'Azure dans un `Ultra` pool de capacité. Trident détecte automatiquement tous les sous-réseaux délégués à Azure NetApp Files dans cet emplacement et place un nouveau volume sur l'un d'eux de manière aléatoire.

```
---
version: 1
storageDriverName: azure-netapp-files
subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
tenantID: 68e4f836-edc1-fake-bff9-b2d865ee56cf
clientID: dd043f63-bf8e-fake-8076-8de91e5713aa
clientSecret: SECRET
location: eastus
serviceLevel: Ultra
capacityPools:
  - application-group-1/account-1/ultra-1
  - application-group-1/account-1/ultra-2
```

Exemple de backend avec des pools de capacité QoS manuels

Cette configuration backend place les volumes dans l'emplacement `eastus` Azure avec des pools de capacité QoS manuels.

```
---
version: 1
storageDriverName: azure-netapp-files
backendName: anf1
location: eastus
labels:
  clusterName: test-cluster-1
  cloud: anf
  nasType: nfs
defaults:
  qosType: Manual
storage:
  - serviceLevel: Ultra
    labels:
      performance: gold
    defaults:
      maxThroughput: 10
  - serviceLevel: Premium
    labels:
      performance: silver
    defaults:
      maxThroughput: 5
  - serviceLevel: Standard
    labels:
      performance: bronze
    defaults:
      maxThroughput: 3
```

Configuration avancée

Cette configuration backend réduit encore la portée du placement des volumes à un seul sous-réseau et modifie également certains paramètres par défaut de provisionnement des volumes.

```
---
version: 1
storageDriverName: azure-netapp-files
subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
tenantID: 68e4f836-edc1-fake-bff9-b2d865ee56cf
clientID: dd043f63-bf8e-fake-8076-8de91e5713aa
clientSecret: SECRET
location: eastus
serviceLevel: Ultra
capacityPools:
  - application-group-1/account-1/ultra-1
  - application-group-1/account-1/ultra-2
virtualNetwork: application-group-1/eastus-prod-vnet
subnet: application-group-1/eastus-prod-vnet/my-subnet
networkFeatures: Standard
nfsMountOptions: vers=3,proto=tcp,timeo=600
limitVolumeSize: 500Gi
defaults:
  exportRule: 10.0.0.0/24,10.0.1.0/24,10.0.2.100
  snapshotDir: "true"
  size: 200Gi
  unixPermissions: "0777"
```

Configuration du pool virtuel

Cette configuration backend définit plusieurs pools de stockage dans un seul fichier. Ceci est utile lorsque vous disposez de plusieurs pools de capacité prenant en charge différents niveaux de service et que vous souhaitez créer des classes de stockage dans Kubernetes pour les représenter. Des étiquettes de pool virtuel ont été utilisées pour différencier les pools en fonction de `performance`.

```
---
version: 1
storageDriverName: azure-netapp-files
subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
tenantID: 68e4f836-edc1-fake-bff9-b2d865ee56cf
clientID: dd043f63-bf8e-fake-8076-8de91e5713aa
clientSecret: SECRET
location: eastus
resourceGroups:
  - application-group-1
networkFeatures: Basic
nfsMountOptions: vers=3,proto=tcp,timeo=600
labels:
  cloud: azure
storage:
  - labels:
      performance: gold
    serviceLevel: Ultra
    capacityPools:
      - application-group-1/netapp-account-1/ultra-1
      - application-group-1/netapp-account-1/ultra-2
    networkFeatures: Standard
  - labels:
      performance: silver
    serviceLevel: Premium
    capacityPools:
      - application-group-1/netapp-account-1/premium-1
  - labels:
      performance: bronze
    serviceLevel: Standard
    capacityPools:
      - application-group-1/netapp-account-1/standard-1
      - application-group-1/netapp-account-1/standard-2
```

Configuration des topologies prises en charge

Trident facilite le provisionnement des volumes pour les charges de travail en fonction des régions et des zones de disponibilité. Le `supportedTopologies` bloc dans cette configuration backend est utilisé pour fournir une liste de régions et de zones par backend. Les valeurs de région et de zone spécifiées ici doivent correspondre aux valeurs de région et de zone des étiquettes sur chaque nœud de cluster Kubernetes. Ces régions et zones représentent la liste des valeurs autorisées pouvant être fournies dans une classe de stockage. Pour les classes de stockage qui contiennent un sous-ensemble des régions et zones fournies dans un backend, Trident crée les volumes dans la région et la zone mentionnées. Pour plus d'informations, consultez "Utiliser la topologie CSI".

```
---
version: 1
storageDriverName: azure-netapp-files
subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
tenantID: 68e4f836-edc1-fake-bff9-b2d865ee56cf
clientID: dd043f63-bf8e-fake-8076-8de91e5713aa
clientSecret: SECRET
location: eastus
serviceLevel: Ultra
capacityPools:
  - application-group-1/account-1/ultra-1
  - application-group-1/account-1/ultra-2
supportedTopologies:
  - topology.kubernetes.io/region: eastus
    topology.kubernetes.io/zone: eastus-1
  - topology.kubernetes.io/region: eastus
    topology.kubernetes.io/zone: eastus-2
```

Définitions des classes de stockage

Les `StorageClass` définitions suivantes se rapportent aux pools de stockage ci-dessus.

Exemple de définitions utilisant `parameter.selector` field

En utilisant `parameter.selector`, vous pouvez spécifier pour chaque `StorageClass` le pool virtuel utilisé pour héberger un volume. Le volume aura les aspects définis dans le pool choisi.

```
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gold
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=gold
allowVolumeExpansion: true

---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: silver
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=silver
allowVolumeExpansion: true

---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: bronze
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=bronze
allowVolumeExpansion: true
```

Exemples de définitions pour les volumes SMB

En utilisant `nasType`, `node-stage-secret-name` et `node-stage-secret-namespace`, vous pouvez spécifier un volume SMB et fournir les informations d'identification Active Directory requises.

Configuration de base sur l'espace de noms par défaut

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: anf-sc-smb
provisioner: csi.trident.netapp.io
parameters:
  backendType: "azure-netapp-files"
  trident.netapp.io/nasType: "smb"
  csi.storage.k8s.io/node-stage-secret-name: "smbcreds"
  csi.storage.k8s.io/node-stage-secret-namespace: "default"
```

Utilisation de secrets différents par espace de noms

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: anf-sc-smb
provisioner: csi.trident.netapp.io
parameters:
  backendType: "azure-netapp-files"
  trident.netapp.io/nasType: "smb"
  csi.storage.k8s.io/node-stage-secret-name: "smbcreds"
  csi.storage.k8s.io/node-stage-secret-namespace: ${pvc.namespace}
```

Utilisation de secrets différents par volume

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: anf-sc-smb
provisioner: csi.trident.netapp.io
parameters:
  backendType: "azure-netapp-files"
  trident.netapp.io/nasType: "smb"
  csi.storage.k8s.io/node-stage-secret-name: ${pvc.name}
  csi.storage.k8s.io/node-stage-secret-namespace: ${pvc.namespace}
```

`nasType: smb`Filtres pour les pools prenant en charge les volumes SMB. `nasType: nfs` ou `nasType: null`filtres pour les pools NFS.

Créer le backend

Après avoir créé le fichier de configuration, exécutez la commande suivante :

```
tridentctl create backend -f <backend-file>
```

Si vous utilisez un cloud Azure non commercial, assurez-vous que `tridentctl` est configuré pour utiliser Azure Resource Manager et les points de terminaison d'authentification de votre environnement cloud. Si la création du backend échoue, vérifiez la configuration de votre backend et consultez les journaux pour en déterminer la cause :

```
tridentctl logs
```

Après avoir identifié et corrigé le problème avec le fichier de configuration, vous pouvez exécuter à nouveau la commande create.

Google Cloud NetApp Volumes

Configurer Google Cloud NetApp Volumes

Vous pouvez configurer Google Cloud NetApp Volumes comme backend pour que Trident puisse provisionner du stockage pour les charges de travail Kubernetes.

Aperçu

Trident prend en charge Google Cloud NetApp Volumes pour les charges de travail NAS (NFS et SMB) et bloc (iSCSI).

- Les charges de travail NAS utilisent le `google-cloud-netapp-volumes` backend
- Les charges de travail par blocs (iSCSI) utilisent le `google-cloud-netapp-volumes-san` backend

Les volumes NAS offrent un stockage de fichiers et sont accessibles via les protocoles NFS ou SMB. Ces volumes prennent en charge l'accès partagé entre plusieurs pods ou nœuds.

Les volumes de blocs offrent un stockage bloc et sont accessibles comme des périphériques iSCSI connectés aux nœuds Kubernetes. Ces volumes sont utilisés lorsque les applications nécessitent un accès au niveau bloc.

Ceci s'applique aux environnements suivants :

- Trident 26.02 et versions ultérieures
- Google Kubernetes Engine (GKE) ou Red Hat OpenShift
- Pools de stockage Google Cloud NetApp Volumes

Pour configurer le stockage bloc (iSCSI), voir "Configurer le stockage bloc (iSCSI)".

Préparez-vous à configurer

L'identité cloud permet aux charges de travail Kubernetes d'accéder aux ressources Google Cloud en s'authentifiant en tant qu'identité de charge de travail au lieu d'utiliser des informations d'identification statiques.

Pour utiliser l'identité cloud avec Google Cloud NetApp Volumes, vous devez disposer des éléments suivants :

- Un cluster Kubernetes déployé à l'aide de Google Kubernetes Engine (GKE)
- L'identité de la charge de travail est activée sur le cluster GKE et le serveur de métadonnées est activé sur les pools de nœuds
- Un compte de service Google Cloud avec le rôle d'administrateur Google Cloud NetApp Volumes (`roles/netapp.admin`) ou un rôle personnalisé équivalent
- Trident installé avec le fournisseur de cloud défini sur `GCP` et l'annotation d'identité cloud configurée

Opérateur Trident

Pour installer Trident à l'aide de l'opérateur Trident, modifiez `tridentorchestrator_cr.yaml` :

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  namespace: trident
  cloudProvider: "GCP"
  cloudIdentity: "iam.gke.io/gcp-service-account: cloudvolumes-admin-
sa@mygcpproject.iam.gserviceaccount.com"
```

Helm

Configurez le fournisseur de cloud et l'identité cloud lors de l'installation de Trident avec Helm :

```
helm install trident trident-operator-100.6.0.tgz \
  --set cloudProvider=GCP \
  --set cloudIdentity="iam.gke.io/gcp-service-account: cloudvolumes-
admin-sa@mygcpproject.iam.gserviceaccount.com"
```

tridentctl

Installez Trident en spécifiant le fournisseur de cloud et l'identité cloud :

```
tridentctl install \
  --cloud-provider=GCP \
  --cloud-identity="iam.gke.io/gcp-service-account: cloudvolumes-admin-
sa@mygcpproject.iam.gserviceaccount.com" \
  -n trident
```

Configurer le stockage NAS

Pour les pools de stockage UNIFIED de Google Cloud NetApp Volumes, Trident applique des règles de nommage et de validation spécifiques à UNIFIED lors des opérations sur les volumes.

Lors de la localisation d'un volume, Trident peut évaluer plusieurs variantes de nom de volume compatibles (par exemple, les formats avec tiret et trait de soulignement) afin d'améliorer la fiabilité de l'importation et de la découverte.

Détails du pilote

Trident fournit le `google-cloud-netapp-volumes` driver pour provisionner le stockage NAS à partir de Google Cloud NetApp Volumes.

Le pilote prend en charge les modes d'accès suivants :

- ReadWriteOnce (RWO)
- ReadOnlyMany (ROX)
- ReadWriteMany (RWX)
- ReadWriteOncePod (RWOP)

Pilote	Protocole	volumeMode	Modes d'accès pris en charge	Systèmes de fichiers pris en charge
`google-cloud-netapp-volumes`	NFS SMB	Système de fichiers	RWO, ROX, RWX, RWOP	`nfs`, `smb`

Configurer un backend NAS Trident

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: gcnv-nas
  namespace: trident
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: "<project-number>"
  location: "<region>"
  sdkTimeout: "600"
  storage:
  - labels:
      cloud: gcp
    network: "<vpc-network>"
```

Provisionner des volumes NAS

Les volumes NAS sont provisionnés à l'aide du `google-cloud-netapp-volumes` backend et prennent en charge les protocoles NFS et SMB.

StorageClass for les volumes NFS

Pour provisionner des volumes NFS, définissez `nasType` sur `nfs`.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gcnv-nfs
provisioner: csi.trident.netapp.io
parameters:
  backendType: "google-cloud-netapp-volumes"
  trident.netapp.io/nasType: "nfs"
allowVolumeExpansion: true
```

StorageClass for les volumes SMB

Pour provisionner des volumes SMB, configurez `nasType` sur `smb` et fournissez les informations d'identification.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gcnv-smb
provisioner: csi.trident.netapp.io
parameters:
  backendType: "google-cloud-netapp-volumes"
  trident.netapp.io/nasType: "smb"
  csi.storage.k8s.io/node-stage-secret-name: "smbcreds"
  csi.storage.k8s.io/node-stage-secret-namespace: "default"
allowVolumeExpansion: true
```

Exemple de PersistentVolumeClaim (RWX)

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: gcnv-nas-rwx
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 100Gi
  storageClassName: gcnv-nfs
```

Exemple de PersistentVolumeClaim (RWO)

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: gcnv-nas-rwo
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 100Gi
  storageClassName: gcnv-nfs
```

Les volumes NAS utilisent `volumeMode: Filesystem`.

Configurer Google Cloud NetApp Volumes pour les charges de travail SAN

Vous pouvez configurer Trident pour provisionner des volumes de stockage bloc à l'aide du protocole iSCSI depuis Google Cloud NetApp Volumes. Les volumes SAN sont provisionnés à partir de pools de stockage Flex Unified en utilisant le `google-cloud-netapp-volumes-san` storage driver.

Ce pilote est dédié aux charges de travail bloc et ne prend pas en charge les protocoles NAS.

Le `google-cloud-netapp-volumes-san` backend est requis pour provisionner des volumes de blocs iSCSI. Le `google-cloud-netapp-volumes` backend prend uniquement en charge les protocoles NAS et ne peut pas être utilisé pour les charges de travail SAN.

Aperçu

Trident prend en charge les charges de travail Google Cloud NetApp Volumes SAN (iSCSI) à l'aide du `google-cloud-netapp-volumes-san` driver.

Les volumes SAN sont provisionnés à partir de pools de stockage Flex Unified et présentés aux nœuds Kubernetes en tant que périphériques de blocs iSCSI.

Ceci s'applique aux environnements suivants :

- Trident 26.02 et versions ultérieures
- Google Kubernetes Engine (GKE) ou Red Hat OpenShift
- Google Cloud NetApp Volumes Flex Pools de stockage unifiés
- Charges de travail basées sur iSCSI

Pools de stockage Flex Unified

Les pools de stockage Flex Unified fournissent du stockage bloc utilisant le protocole iSCSI et sont nécessaires au provisionnement SAN :

- Les pools régionaux Flex Unified sont pris en charge.
- Les pools ZONAL Flex Unified sont pris en charge à partir de Trident 26.02.1.
- Seul le niveau de service **Flex** est pris en charge pour les charges de travail SAN.

Configurer un backend SAN Trident

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: gcnv-san
  namespace: trident
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes-san
  projectNumber: "<project-number>"
  location: "<region>"
  sdkTimeout: "600"
  storage:
  - labels:
      cloud: gcp
      performance: flex
    network: "<vpc-network>"
    serviceLevel: Flex
```

Créer un StorageClass

Après avoir configuré le backend SAN, créez un StorageClass qui référence le `google-cloud-netapp-volumes-san` driver.

Le type de système de fichiers est défini dans le StorageClass, pas dans le backend.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gcnv-san
provisioner: csi.trident.netapp.io
parameters:
  backendType: "google-cloud-netapp-volumes-san"
  fsType: "ext4"
allowVolumeExpansion: true
```

Types de systèmes de fichiers pris en charge :

- `ext4` (défaut)
- `ext3`
- `xfs`

> Le pilote SAN ne prend en charge que le niveau de service Flex et n'utilise pas les paramètres backend spécifiques au NAS tels que `exportRule`, `unixPermissions`, `nasType`, `snapshotDir`, `nfsMountOptions` ou les paramètres liés à la hiérarchisation.

Approvisionner des volumes de blocs

ReadWriteOnce (RWO)

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: gcnv-san-rwo
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 100Gi
  storageClassName: gcnv-san
```

ReadWriteOncePod (RWOP)

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: gcnv-san-rwop
spec:
  accessModes:
    - ReadWriteOncePod
  resources:
    requests:
      storage: 100Gi
  storageClassName: gcnv-san
```

ReadOnlyMany (ROX)

Une méthode courante pour ROX consiste à cloner un volume ReadWriteOnce existant et à monter le clone en lecture seule.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: gcnv-san-rox
spec:
  accessModes:
    - ReadOnlyMany
  resources:
    requests:
      storage: 100Gi
  storageClassName: gcnv-san
  dataSource:
    kind: PersistentVolumeClaim
    name: gcnv-san-rwo
```

ReadWriteMany (RWX) — bloc brut uniquement

ReadWriteMany n'est pris en charge que lorsque `volumeMode: Block`.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: gcnv-san-raw-rwx
spec:
  accessModes:
    - ReadWriteMany
  volumeMode: Block
  resources:
    requests:
      storage: 100Gi
  storageClassName: gcnv-san
```

Comportement du volume de bloc

Les volumes de blocs sont provisionnés en tant que LUN iSCSI et présentés aux nœuds Kubernetes en tant que périphériques de blocs.

Volumes de blocs :

- Utilisez le protocole iSCSI
- Prise en charge du système de fichiers et de la présentation des blocs bruts
- Sont rattachés et gérés par Trident
- Prise en charge de plusieurs modes d'accès Kubernetes

Modes d’accès

Les volumes de blocs provisionnés par Trident prennent en charge les modes d’accès suivants :

- `ReadWriteOnce` (RWO)
- `ReadOnlyMany` (ROX)
- `ReadWriteOncePod` (RWOP)
- `ReadWriteMany` (RWX), pris en charge uniquement lorsque `volumeMode: Block`

Comportement de volumeMode

Le `volumeMode`champ contrôle la façon dont un volume de bloc est exposé :

- `Filesystem` Trident formate et monte le volume.
- `Block` Trident attache le périphérique et l’expose comme un périphérique de bloc brut.

Opérations prises en charge

Volumes de blocs provisionnés à l’aide du pilote `google-cloud-netapp-volumes-san` prennent en charge :

- Créer
- Supprimer
- Cloner
- Instantané
- Redimensionner
- Importer

Comportement de surprovisionnement supplémentaire de GiB

Google Cloud NetApp Volumes block volumes incluent des métadonnées internes. Cette surcharge réduit la taille du périphérique visible par le noyau par rapport à la capacité provisionnée.

Les tests montrent :

- Environ 300 Kio de surcharge lors de la création initiale
- Jusqu’à environ 107 MiB de surcharge après un redimensionnement

Étant donné que Google Cloud NetApp Volumes n’accepte que des allocations de Gio entiers, Trident garantit que la taille utilisable du périphérique respecte ou dépasse toujours la demande PVC en :

- Arrondi de la taille demandée au Gio entier supérieur
- Ajout d’un tampon supplémentaire de 1 Gio

Exemple :

- Requête PVC : 100 Gio
- Taille provisionnée dans Google Cloud NetApp Volumes : 101 GiB

- Espace utilisable visible par l'application : au moins 100 GiB

Exemples de pods

Volume de bloc monté sur le système de fichiers (RWO)

```
apiVersion: v1
kind: Pod
metadata:
  name: app-rwo
spec:
  containers:
  - name: app
    image: ubuntu:22.04
    command: ["sleep", "infinity"]
    volumeMounts:
    - name: data
      mountPath: /mnt/data
  volumes:
  - name: data
    persistentVolumeClaim:
      claimName: gcnv-san-rwo
```

Périphérique de bloc brut (RWX)

```
apiVersion: v1
kind: Pod
metadata:
  name: app-raw-rwx
spec:
  containers:
  - name: app
    image: ubuntu:22.04
    command: ["sleep", "infinity"]
    volumeDevices:
    - name: data
      devicePath: /dev/xda
  volumes:
  - name: data
    persistentVolumeClaim:
      claimName: gcnv-san-raw-rwx
```

Comportement d'attachement et de montage

Pour les volumes SAN provisionnés à partir de Google Cloud NetApp Volumes :

- Trident crée un Logical Unit Number (LUN) dans un pool de stockage Flex Unified.
- Lors de la publication, Trident associe le LUN à un groupe d'hôtes par nœud.
- Lors de la mise en scène des nœuds, Trident :
  - Se connecte à la cible iSCSI
  - Découvre le LUN
  - Configure le multipath
- Si `volumeMode: Filesystem`Trident le nécessite, Trident formate le périphérique et le monte.
- Si `volumeMode: Block` Trident attache le périphérique et l'expose directement au pod sans le formater ni le monter.

> Les volumes de blocs SAN ne prennent pas en charge le verrouillage distribué ni la coordination des écritures. Lorsqu'un volume de blocs est accédé par plusieurs nœuds (ReadWriteMany avec `volumeMode: Block`), l'application ou le système de fichiers doit gérer la concurrence.

Préparez-vous à configurer un backend Google Cloud NetApp Volumes

Avant de pouvoir configurer votre backend Google Cloud NetApp Volumes, vous devez vous assurer que les conditions suivantes sont remplies.

Prérequis pour les volumes NFS ou SMB

Si vous utilisez Google Cloud NetApp Volumes pour la première fois ou dans un nouvel emplacement, une configuration initiale est nécessaire pour configurer Google Cloud NetApp Volumes et créer un volume NFS ou SMB. Consultez "Avant de commencer".

Assurez-vous de disposer des éléments suivants avant de configurer Google Cloud NetApp Volumes backend :

- Un compte Google Cloud configuré avec le service Google Cloud NetApp Volumes. Consultez "Google Cloud NetApp Volumes".
- Numéro de projet de votre compte Google Cloud. Consultez "Identification des projets".
- Un compte de service Google Cloud avec le rôle NetApp Volumes Admin (`roles/netapp.admin`. Consultez "Rôles et autorisations d'Identity and Access Management".
- Fichier de clé API pour votre compte GCNV. Consultez "Créer une clé de compte de service"
- Un pool de stockage. Consultez "Aperçu des pools de stockage".

Pour plus d'informations sur la configuration de l'accès à Google Cloud NetApp Volumes, consultez "Configurer l'accès à Google Cloud NetApp Volumes".

Options et exemples de configuration du backend Google Cloud NetApp Volumes

Découvrez les options de configuration backend pour Google Cloud NetApp Volumes et consultez des exemples de configuration.

Options de configuration du backend

Chaque backend provisionne des volumes dans une seule région Google Cloud. Pour créer des volumes dans d'autres régions, vous pouvez définir des backends supplémentaires.

Paramètre	Description	Défaut
`version`		Toujours 1
`storageDriverName`	Nom du pilote de stockage	La valeur de `storageDriverName` doit être spécifiée comme « google-cloud-netapp-volumes ».
`backendName`	(Facultatif) Nom personnalisé du stockage backend	Nom du driver + "_" + partie de la clé API
`storagePools`	Paramètre optionnel utilisé pour spécifier les pools de stockage pour la création de volumes.	
`projectNumber`	Numéro de projet du compte Google Cloud. La valeur se trouve sur la page d'accueil du portail Google Cloud.	
`location`	L'emplacement Google Cloud où Trident crée les volumes GCNV. Lors de la création de clusters Kubernetes interrégionaux, les volumes créés dans un `location` peuvent être utilisés dans des charges de travail planifiées sur des nœuds répartis dans plusieurs régions Google Cloud. Le trafic interrégional engendre des coûts supplémentaires.	
`apiKey`	Clé API pour le compte de service Google Cloud avec le `netapp.admin` rôle. Elle inclut le contenu au format JSON du fichier de clé privée d'un compte de service Google Cloud (copié tel quel dans le fichier de configuration du backend). Le `apiKey` doit inclure des paires clé-valeur pour les clés suivantes : `type`, `project_id`, `client_email`, `client_id`, `auth_uri`, `token_uri`, `auth_provider_x509_cert_url` et `client_x509_cert_url`.	
`nfsMountOptions`	Contrôle précis des options de montage NFS.	"nfsvers=3"
`limitVolumeSize`	L'approvisionnement échoue si la taille du volume demandée est supérieure à cette valeur.	"" (non appliqué par défaut)
`serviceLevel`	Le niveau de service d'un pool de stockage et de ses volumes. Les valeurs sont `flex`, `standard`, `premium` ou `extreme`.	
`labels`	Ensemble d'étiquettes arbitraires au format JSON à appliquer aux volumes	""
`network`	Réseau Google Cloud utilisé pour les volumes GCNV.	
`debugTraceFlags`	Options de débogage à utiliser lors du dépannage. Exemple, `{"api":false, "method":true}`. N'utilisez cette option que si vous effectuez un dépannage et avez besoin d'un journal détaillé.	null

Paramètre	Description	Défaut
`nasType`	Configurez la création de volumes NFS ou SMB. Les options sont `nfs`, `smb` ou null. La valeur null correspond par défaut à des volumes NFS.	`nfs`
`supportedTopologies`	Représente une liste de régions et de zones prises en charge par ce backend. Pour plus d'informations, consultez "Utiliser la topologie CSI". Par exemple : `supportedTopologies: - topology.kubernetes.io/region: asia-east1 topology.kubernetes.io/zone: asia-east1-a`	

Options de provisionnement de volume

Vous pouvez contrôler le provisionnement des volumes par défaut dans la section `defaults` du fichier de configuration.

Paramètre	Description	Défaut
`exportRule`	Les règles d'exportation pour les nouveaux volumes. Doit être une liste séparée par des virgules de toute combinaison d'adresses IPv4.	"0.0.0.0/0"
`snapshotDir`	Accès au `.snapshot` répertoire	`true`, `false` (Le comportement par défaut peut varier. À définir explicitement) « false » pour NFSv3
`snapshotReserve`	Pourcentage du volume réservé aux instantanés	"" (accepter la valeur par défaut de 0)
`unixPermissions`	Les permissions Unix des nouveaux volumes (4 chiffres octaux).	""

Exemples de configurations

Les exemples suivants présentent des configurations de base qui laissent la plupart des paramètres par défaut. C'est la manière la plus simple de définir un backend.

Configuration minimale

Il s'agit de la configuration minimale absolue du backend. Avec cette configuration, Trident découvre tous vos pools de stockage délégués à Google Cloud NetApp Volumes à l'emplacement configuré, et place les nouveaux volumes sur l'un de ces pools de façon aléatoire. Parce que `nasType` est omis, la `nfs` configuration par défaut s'applique et le backend provisionnera des volumes NFS.

Cette configuration est idéale lorsque vous débutez avec Google Cloud NetApp Volumes et que vous faites des essais, mais en pratique, vous pourriez avoir besoin de définir une portée supplémentaire pour les volumes que vous provisionnez.

Remplacez `<id_value>` et `<key_value>` par vos identifiants de compte de service.

```
---
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-gcnv-secret
type: Opaque
stringData:
  private_key_id: "<id_value>"
  private_key: |
    -----BEGIN PRIVATE KEY-----
    <key_value>
    -----END PRIVATE KEY-----
---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-gcnv
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: "123455380079"
  location: europe-west6
  serviceLevel: premium
  apiKey:
    type: service_account
    project_id: my-gcnv-project
    client_email: myproject-prod@my-gcnv-
project.iam.gserviceaccount.com
    client_id: "103346282737811234567"
    auth_uri: https://accounts.google.com/o/oauth2/auth
    token_uri: https://oauth2.googleapis.com/token
    auth_provider_x509_cert_url:
https://www.googleapis.com/oauth2/v1/certs
    client_x509_cert_url:
https://www.googleapis.com/robot/v1/metadata/x509/myproject-prod%40my-
gcnv-project.iam.gserviceaccount.com
  credentials:
    name: backend-tbc-gcnv-secret
```

Configuration pour les volumes SMB

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-gcnv1
  namespace: trident
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: "123456789"
  location: asia-east1
  serviceLevel: flex
  nasType: smb
  apiKey:
    type: service_account
    project_id: cloud-native-data
    client_email: trident-sample@cloud-native-
data.iam.gserviceaccount.com
    client_id: "123456789737813416734"
    auth_uri: https://accounts.google.com/o/oauth2/auth
    token_uri: https://oauth2.googleapis.com/token
    auth_provider_x509_cert_url:
https://www.googleapis.com/oauth2/v1/certs
    client_x509_cert_url:
https://www.googleapis.com/robot/v1/metadata/x509/trident-
sample%40cloud-native-data.iam.gserviceaccount.com
  credentials:
    name: backend-tbc-gcnv-secret
```

Configuration avec le filtre StoragePools

```
---
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-gcnv-secret
type: Opaque
stringData:
  private_key_id: "<id_value>"
  private_key: |
    -----BEGIN PRIVATE KEY-----
    <key_value>
    -----END PRIVATE KEY-----
---

---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-gcnv
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: "123455380079"
  location: europe-west6
  serviceLevel: premium
  storagePools:
    - premium-pool1-europe-west6
    - premium-pool2-europe-west6
  apiKey:
    type: service_account
    project_id: my-gcnv-project
    client_email: myproject-prod@my-gcnv-
project.iam.gserviceaccount.com
    client_id: "103346282737811234567"
    auth_uri: https://accounts.google.com/o/oauth2/auth
    token_uri: https://oauth2.googleapis.com/token
    auth_provider_x509_cert_url:
https://www.googleapis.com/oauth2/v1/certs
    client_x509_cert_url:
https://www.googleapis.com/robot/v1/metadata/x509/myproject-prod%40my-
gcnv-project.iam.gserviceaccount.com
  credentials:
    name: backend-tbc-gcnv-secret
```

Configuration du pool virtuel

Cette configuration backend définit plusieurs pools virtuels dans un seul fichier. Les pools virtuels sont définis dans la `storage` section. Ils sont utiles lorsque vous disposez de plusieurs pools de stockage prenant en charge différents niveaux de service et que vous souhaitez créer des classes de stockage dans Kubernetes pour les représenter. Les étiquettes des pools virtuels sont utilisées pour différencier les pools. Par exemple, dans l'exemple ci-dessous, `performance` label et `serviceLevel` type sont utilisés pour différencier les pools virtuels.

Vous pouvez également définir des valeurs par défaut applicables à tous les pools virtuels et remplacer les valeurs par défaut pour des pools virtuels individuels. Dans l'exemple suivant, `snapshotReserve` et `exportRule` servent de valeurs par défaut pour tous les pools virtuels.

Pour plus d'informations, consultez "Pools virtuels".

```
---
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-gcnv-secret
type: Opaque
stringData:
  private_key_id: "<id_value>"
  private_key: |
    -----BEGIN PRIVATE KEY-----
    <key_value>
    -----END PRIVATE KEY-----

---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-gcnv
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: "123455380079"
  location: europe-west6
  apiKey:
    type: service_account
    project_id: my-gcnv-project
    client_email: myproject-prod@my-gcnv-
project.iam.gserviceaccount.com
    client_id: "103346282737811234567"
    auth_uri: https://accounts.google.com/o/oauth2/auth
    token_uri: https://oauth2.googleapis.com/token
    auth_provider_x509_cert_url:
https://www.googleapis.com/oauth2/v1/certs
```

```
    client_x509_cert_url:
https://www.googleapis.com/robot/v1/metadata/x509/myproject-prod%40my-
gcnv-project.iam.gserviceaccount.com
  credentials:
    name: backend-tbc-gcnv-secret
  defaults:
    snapshotReserve: "10"
    exportRule: 10.0.0.0/24
  storage:
    - labels:
        performance: extreme
      serviceLevel: extreme
      defaults:
        snapshotReserve: "5"
        exportRule: 0.0.0.0/0
    - labels:
        performance: premium
      serviceLevel: premium
    - labels:
        performance: standard
      serviceLevel: standard
```

Identité cloud pour GKE

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-gcp-gcnv
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: '012345678901'
  network: gcnv-network
  location: us-west2
  serviceLevel: Premium
  storagePool: pool-premium1
```

Configuration des topologies prises en charge

Trident facilite le provisionnement des volumes pour les charges de travail en fonction des régions et des zones de disponibilité. Le `supportedTopologies` bloc dans cette configuration backend est utilisé pour fournir une liste de régions et de zones par backend. Les valeurs de région et de zone spécifiées ici doivent correspondre aux valeurs de région et de zone des étiquettes sur chaque nœud de cluster Kubernetes. Ces régions et zones représentent la liste des valeurs autorisées pouvant être fournies dans une classe de stockage. Pour les classes de stockage qui contiennent un sous-ensemble des régions et zones fournies dans un backend, Trident crée les volumes dans la région et la zone mentionnées. Pour plus d'informations, consultez "Utiliser la topologie CSI".

```
---
version: 1
storageDriverName: google-cloud-netapp-volumes
subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
tenantID: 68e4f836-edc1-fake-bff9-b2d865ee56cf
clientID: dd043f63-bf8e-fake-8076-8de91e5713aa
clientSecret: SECRET
location: asia-east1
serviceLevel: flex
supportedTopologies:
  - topology.kubernetes.io/region: asia-east1
    topology.kubernetes.io/zone: asia-east1-a
  - topology.kubernetes.io/region: asia-east1
    topology.kubernetes.io/zone: asia-east1-b
```

Et ensuite ?

Après avoir créé le fichier de configuration, exécutez la commande suivante :

```
kubectl create -f <backend-file>
```

Pour vérifier que le backend a bien été créé, exécutez la commande suivante :

```
kubectl get tridentbackendconfig

NAME                BACKEND NAME        BACKEND UUID
PHASE   STATUS
backend-tbc-gcnv    backend-tbc-gcnv    b2fd1ff9-b234-477e-88fd-713913294f65
Bound   Success
```

Si la création du backend échoue, cela signifie qu'il y a un problème avec la configuration du backend. Vous pouvez décrire le backend à l'aide de la commande `kubectl get tridentbackendconfig <backend-name>` ou consulter les journaux pour déterminer la cause en exécutant la commande suivante :

```
tridentctl logs
```

Après avoir identifié et corrigé le problème lié au fichier de configuration, vous pouvez supprimer le backend et exécuter à nouveau la commande de création.

Définitions des classes de stockage

L'exemple suivant est une `StorageClass`définition de base qui fait référence au backend ci-dessus.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gcnv-nfs-sc
provisioner: csi.trident.netapp.io
parameters:
  backendType: "google-cloud-netapp-volumes"
```

Exemples de définitions utilisant le champ `parameter.selector` :

En utilisant `parameter.selector`, vous pouvez spécifier pour chaque `StorageClass` le "pool virtuel" utilisé pour héberger un volume. Le volume aura les aspects définis dans le pool choisi.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: extreme-sc
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=extreme
  backendType: google-cloud-netapp-volumes

---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: premium-sc
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=premium
  backendType: google-cloud-netapp-volumes

---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: standard-sc
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=standard
  backendType: google-cloud-netapp-volumes
```

Pour plus de détails sur les classes de stockage, reportez-vous à "Créer une classe de stockage".

Exemples de définitions pour les volumes SMB

En utilisant `nasType`, `node-stage-secret-name` et `node-stage-secret-namespace`, vous pouvez spécifier un volume SMB et fournir les informations d'identification Active Directory requises. Tout utilisateur/mot de passe Active Directory, avec ou sans autorisations, peut être utilisé pour le secret de l'étape du nœud.

Configuration de base sur l'espace de noms par défaut

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gcnv-sc-smb
provisioner: csi.trident.netapp.io
parameters:
  backendType: "google-cloud-netapp-volumes"
  trident.netapp.io/nasType: "smb"
  csi.storage.k8s.io/node-stage-secret-name: "smbcreds"
  csi.storage.k8s.io/node-stage-secret-namespace: "default"
```

Utilisation de secrets différents par espace de noms

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gcnv-sc-smb
provisioner: csi.trident.netapp.io
parameters:
  backendType: "google-cloud-netapp-volumes"
  trident.netapp.io/nasType: "smb"
  csi.storage.k8s.io/node-stage-secret-name: "smbcreds"
  csi.storage.k8s.io/node-stage-secret-namespace: ${pvc.namespace}
```

Utilisation de secrets différents par volume

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gcnv-sc-smb
provisioner: csi.trident.netapp.io
parameters:
  backendType: "google-cloud-netapp-volumes"
  trident.netapp.io/nasType: "smb"
  csi.storage.k8s.io/node-stage-secret-name: ${pvc.name}
  csi.storage.k8s.io/node-stage-secret-namespace: ${pvc.namespace}
```

`nasType: smb`Filtres pour les pools prenant en charge les volumes SMB. `nasType: nfs ou `nasType: null`filtres pour les pools NFS.

Exemple de définition du PVC

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: gcnv-nfs-pvc
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 100Gi
  storageClassName: gcnv-nfs-sc
```

Pour vérifier si le PVC est lié, exécutez la commande suivante :

```
kubectl get pvc gcnv-nfs-pvc

NAME          STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS AGE
gcnv-nfs-pvc  Bound    pvc-b00f2414-e229-40e6-9b16-ee03eb79a213  100Gi
RWX            gcnv-nfs-sc  1m
```

Configurer la hiérarchisation automatique pour Google Cloud NetApp Volumes

La hiérarchisation automatique est configurée via les paramètres du backend Trident et les annotations PersistentVolumeClaim lors du provisionnement des volumes. Vous pouvez configurer la hiérarchisation automatique pour Google Cloud NetApp Volumes à l'aide de Trident.

Aperçu

La hiérarchisation automatique permet à Trident de provisionner des volumes qui déplacent automatiquement les données inactives d'un niveau de performance vers un niveau de capacité. Cela réduit le coût de stockage tout en préservant les performances pour les données fréquemment consultées.

Trident applique les paramètres de hiérarchisation automatique uniquement lors de la création du volume. Les modifications après le provisionnement ne sont pas prises en charge dans Trident 26.02.

Concepts

hiérarchisation automatique

Le système de hiérarchisation automatique déplace les données rarement consultées d'un niveau de performance vers un niveau de capacité en fonction des modèles d'accès. Le déplacement des données s'effectue de manière asynchrone et n'est pas immédiat.

politique de hiérarchisation

La politique de hiérarchisation détermine si la hiérarchisation automatique est activée pour un volume.

Les politiques suivantes sont prises en charge : * `auto` : Active la hiérarchisation automatique en fonction des modèles d'accès * `none` : Désactive la hiérarchisation automatique

Jours de refroidissement

Les jours de refroidissement spécifient le nombre minimal de jours pendant lesquels un bloc de données doit rester inactif avant de pouvoir être classé par niveau. Les jours de refroidissement s'appliquent uniquement lorsque la politique de classement par niveau est définie sur `auto`.

Modèle de configuration

Étendues de configuration

La hiérarchisation automatique peut être configurée à plusieurs niveaux :

- **Portée du pool de stockage** S'applique à tous les volumes provisionnés à partir du pool.
- **Portée du volume** S'applique à un seul volume par le biais des annotations PersistentVolumeClaim.

Trident détermine la configuration effective en fonction de l'endroit où chaque paramètre est défini.

Précédence de configuration

Lorsque le même paramètre est défini à plusieurs niveaux, Trident applique l'ordre de priorité suivant :

1. Annotations PersistentVolumeClaim
2. Configuration du backend Trident
3. Paramètres par défaut du storage pool

Les paramètres définis à un niveau de priorité supérieur remplacent les valeurs de niveau inférieur.

fonctionnalité prise en charge dans Trident 26.02

Trident 26.02 prend en charge les fonctionnalités de hiérarchisation automatique suivantes pour Google Cloud NetApp Volumes :

- Activation ou désactivation du auto-tiering lors de l'allocation des volumes
- Définition d'une politique de hiérarchisation dans la configuration du backend Trident
- Remplacement de la politique de hiérarchisation et du nombre de jours de refroidissement par volume à l'aide d'annotations PVC
- Configuration des jours de refroidissement pour les volumes avec auto-tiering activé

fonctionnalité non prise en charge dans Trident 26.02

Les opérations suivantes ne sont pas prises en charge :

- Modification des paramètres de hiérarchisation automatique après la création du volume
- Modification des politiques de hiérarchisation sur les volumes existants à l'aide des mises à jour Kubernetes
- Application des paramètres de hiérarchisation automatique en dehors des workflows de provisionnement gérés par Trident

Paramètres de configuration du backend

Les paramètres suivants contrôlent le comportement de hiérarchisation automatique lorsqu'ils sont définis dans la configuration du backend Trident :

Paramètre	Obligatoire	Description
tieringPolicy	Non	Politique de hiérarchisation des volumes (`auto` ou `none`)
tieringMinimumCoolingDays	Non	Nombre de jours d'inactivité avant le classement des données (plage : 2–183, valeur par défaut : 31)

Remplacements au niveau du volume à l'aide des annotations PersistentVolumeClaim

Annotations prises en charge

Les annotations PersistentVolumeClaim permettent de remplacer, volume par volume, les paramètres de hiérarchisation automatique.

Annotation	Description
`trident.netapp.io/tieringPolicy`	Remplace la politique de hiérarchisation pour le volume
`trident.netapp.io/tieringMinimumCoolingDays`	Remplace la valeur des jours de refroidissement pour le volume

Exemple : PersistentVolumeClaim avec des substitutions de hiérarchisation automatique

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: auto-tiering-pvc
  annotations:
    trident.netapp.io/tieringPolicy: auto
    trident.netapp.io/tieringMinimumCoolingDays: "45"
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: google-cloud-netapp-volumes-auto-tiering
  resources:
    requests:
      storage: 500Gi
```

Comportement et limitations

Comportement de provisionnement

- Les paramètres de hiérarchisation automatique sont évalués et appliqués uniquement lors de la création du volume.
- Trident ne réconcilie pas la configuration de hiérarchisation après le provisionnement.
- Les jours de refroidissement sont ignorés lorsque la politique de hiérarchisation est définie sur `none`.

Limitations de la plateforme

- La hiérarchisation automatique est prise en charge uniquement pour les volumes NAS (NFS et SMB).
- Les volumes de blocs (iSCSI) ne prennent pas en charge l'auto-tiering.
- Le pool de stockage Google Cloud NetApp Volumes doit avoir la hiérarchisation automatique activée dans Google Cloud.

Valeurs prises en charge

- Plage de valeurs valide `tieringMinimumCoolingDays`: 2 à 183
- Valeur par défaut : 31

Configurer un backend NetApp HCI ou SolidFire

Apprenez à créer et à utiliser un backend Element avec votre installation Trident.

Détails du pilote Element

Trident fournit le `solidfire-san` pilote de stockage permettant de communiquer avec le cluster. Les modes d'accès pris en charge sont : *ReadWriteOnce* (RWO), *ReadOnlyMany* (ROX), *ReadWriteMany* (RWX), *ReadWriteOncePod* (RWOP).

Le `solidfire-san` pilote de stockage prend en charge les modes de volume *file* et *block*. Pour le

`Filesystem` volumeMode, Trident crée un volume et crée un système de fichiers. Le type de système de fichiers est spécifié par le StorageClass.

Pilote	Protocole	VolumeMode	Modes d'accès pris en charge	Systèmes de fichiers pris en charge
`solidfire-san`	iSCSI	Bloc	RWO, ROX, RWX, RWOP	Aucun système de fichiers. Périphérique de bloc brut.
`solidfire-san`	iSCSI	Système de fichiers	RWO, RWOP	`xfs`, `ext3`, `ext4`

Avant de commencer

Vous aurez besoin des éléments suivants avant de créer un backend Element.

- Un système de stockage pris en charge qui exécute le logiciel Element.
- Identifiants d'un administrateur de cluster NetApp HCI/SolidFire ou d'un utilisateur locataire pouvant gérer les volumes.
- Tous vos nœuds de travail Kubernetes doivent disposer des outils iSCSI appropriés. Consultez "Informations de préparation du nœud de travail".

Options de configuration du backend

Consultez le tableau suivant pour les options de configuration du backend :

Paramètre	Description	Défaut
`version`		Toujours 1
`storageDriverName`	Nom du pilote de stockage	Toujours "solidfire-san"
`backendName`	Nom personnalisé ou le stockage backend	"solidfire_" + adresse IP de stockage (iSCSI)
`Endpoint`	MVIP pour le cluster SolidFire avec les identifiants du locataire	
`SVIP`	Adresse IP et port de stockage (iSCSI)	
`labels`	Ensemble d'étiquettes au format JSON arbitraires à appliquer aux volumes.	""
`TenantName`	Nom du tenant à utiliser (créé s'il n'est pas trouvé)	
`InitiatorIFace`	Limiter le trafic iSCSI à une interface d'hôte spécifique	"default"
`UseCHAP`	Utilisez CHAP pour authentifier iSCSI. Trident utilise CHAP.	true

Paramètre	Description	Défaut
`AccessGroups`	Liste des ID de groupes d'accès à utiliser	Recherche l'ID d'un groupe d'accès nommé "trident"
`Types`	Spécifications QoS	
`limitVolumeSize`	Échec de l'approvisionnement si la taille du volume demandée est supérieure à cette valeur	"" (non appliqué par défaut)
`debugTraceFlags`	Options de débogage à utiliser lors du dépannage. Exemple, {"api":false, "method":true}	null

AVERTISSEMENT	N'utilisez pas `debugTraceFlags` sauf si vous effectuez un dépannage et avez besoin d'un vidage détaillé du journal.

Exemple 1 : Configuration du backend pour `solidfire-san` driver avec trois types de volumes

Cet exemple illustre un fichier backend utilisant l'authentification CHAP et modélisant trois types de volumes avec des garanties QoS spécifiques. Vous définirez ensuite probablement des classes de stockage pour utiliser chacune d'elles à l'aide du `IOPS` storage class parameter.

```
---
version: 1
storageDriverName: solidfire-san
Endpoint: https://<user>:<password>@<mvip>/json-rpc/8.0
SVIP: <svip>:3260
TenantName: <tenant>
labels:
  k8scluster: dev1
  backend: dev1-element-cluster
UseCHAP: true
Types:
  - Type: Bronze
    Qos:
      minIOPS: 1000
      maxIOPS: 2000
      burstIOPS: 4000
  - Type: Silver
    Qos:
      minIOPS: 4000
      maxIOPS: 6000
      burstIOPS: 8000
  - Type: Gold
    Qos:
      minIOPS: 6000
      maxIOPS: 8000
      burstIOPS: 10000
```

Exemple 2 : Configuration du backend et de la classe de stockage pour `solidfire-san` driver avec pools virtuels

Cet exemple montre le fichier de définition du backend configuré avec des pools virtuels ainsi que StorageClasses qui y font référence.

Trident copie les étiquettes présentes sur un pool de stockage vers le LUN de stockage principal lors du provisionnement. Pour plus de simplicité, les administrateurs de stockage peuvent définir des étiquettes par pool virtuel et regrouper les volumes par étiquette.

Dans l'exemple de fichier de définition de backend ci-dessous, des valeurs par défaut spécifiques sont définies pour tous les pools de stockage, qui définissent le `type` à Silver. Les pools virtuels sont définis dans la section `storage`. Dans cet exemple, certains pools de stockage définissent leur propre type, et certains pools remplacent les valeurs par défaut définies ci-dessus.

```
---
version: 1
storageDriverName: solidfire-san
Endpoint: https://<user>:<password>@<mvip>/json-rpc/8.0
```

```
SVIP: <svip>:3260
TenantName: <tenant>
UseCHAP: true
Types:
  - Type: Bronze
    Qos:
      minIOPS: 1000
      maxIOPS: 2000
      burstIOPS: 4000
  - Type: Silver
    Qos:
      minIOPS: 4000
      maxIOPS: 6000
      burstIOPS: 8000
  - Type: Gold
    Qos:
      minIOPS: 6000
      maxIOPS: 8000
      burstIOPS: 10000
type: Silver
labels:
  store: solidfire
  k8scluster: dev-1-cluster
region: us-east-1
storage:
  - labels:
      performance: gold
      cost: "4"
    zone: us-east-1a
    type: Gold
  - labels:
      performance: silver
      cost: "3"
    zone: us-east-1b
    type: Silver
  - labels:
      performance: bronze
      cost: "2"
    zone: us-east-1c
    type: Bronze
  - labels:
      performance: silver
      cost: "1"
    zone: us-east-1d
```

Les définitions StorageClass suivantes se rapportent aux pools virtuels mentionnés ci-dessus. En utilisant le

champ `parameters.selector`, chaque StorageClass indique le ou les pools virtuels pouvant héberger un volume. Le volume possédera les aspects définis dans le pool virtuel choisi.

Le premier StorageClass (`solidfire-gold-four`) correspond au premier pool virtuel. Il s'agit du seul pool offrant des performances gold avec un `Volume Type QoS` de Gold. Le dernier StorageClass (`solidfire-silver`) fait référence à tout pool de stockage offrant des performances silver. Trident déterminera quel pool virtuel est sélectionné et s'assurera que l'exigence de stockage est respectée.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: solidfire-gold-four
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=gold; cost=4
  fsType: ext4

---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: solidfire-silver-three
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=silver; cost=3
  fsType: ext4

---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: solidfire-bronze-two
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=bronze; cost=2
  fsType: ext4

---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: solidfire-silver-one
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=silver; cost=1
  fsType: ext4
```

```
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: solidfire-silver
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=silver
  fsType: ext4
```

Pour plus d'informations

- "Groupes d'accès au volume"

Pilotes ONTAP SAN

Présentation du pilote ONTAP SAN

Découvrez comment configurer un backend ONTAP avec les pilotes SAN ONTAP et Cloud Volumes ONTAP.

Détails du pilote ONTAP SAN

Trident fournit les pilotes de stockage SAN suivants pour communiquer avec le cluster ONTAP. Les modes d'accès pris en charge sont : *ReadWriteOnce* (RWO), *ReadOnlyMany* (ROX), *ReadWriteMany* (RWX), *ReadWriteOncePod* (RWOP).

Pilote	Protocole	volumeMode	Modes d'accès pris en charge	Systèmes de fichiers pris en charge
`ontap-san`	iSCSI SCSI sur FC	Bloc	RWO, ROX, RWX, RWOP	Aucun système de fichiers ; périphérique bloc brut
`ontap-san`	iSCSI SCSI sur FC	Système de fichiers	RWO, RWOP ROX et RWX ne sont pas disponibles en mode volume du système de fichiers.	`xfs`, `ext3`, `ext4`
`ontap-san`	NVMe/TCP Consultez Considérations supplémentaires pour NVMe/TCP.	Bloc	RWO, ROX, RWX, RWOP	Aucun système de fichiers ; périphérique bloc brut

Pilote	Protocole	volumeMode	Modes d'accès pris en charge	Systèmes de fichiers pris en charge
`ontap-san`	NVMe/TCP Consultez Considérations supplémentaires pour NVMe/TCP.	Système de fichiers	RWO, RWOP ROX et RWX ne sont pas disponibles en mode volume du système de fichiers.	`xfs`, `ext3`, `ext4`
`ontap-san-economy`	iSCSI	Bloc	RWO, ROX, RWX, RWOP	Aucun système de fichiers ; périphérique bloc brut
`ontap-san-economy`	iSCSI	Système de fichiers	RWO, RWOP ROX et RWX ne sont pas disponibles en mode volume du système de fichiers.	`xfs`, `ext3`, `ext4`

AVERTISSEMENT

- Utilisez `ontap-san-economy` uniquement si le nombre d'utilisations du volume persistant doit être supérieur à "limites de volume ONTAP prises en charge".
- Utilisez `ontap-nas-economy` uniquement si le nombre d'utilisations de volumes persistants est censé être supérieur à "limites de volume ONTAP prises en charge" et que le pilote `ontap-san-economy` ne peut pas être utilisé.
- N'utilisez pas `ontap-nas-economy` si vous prévoyez un besoin de protection des données, de reprise après sinistre ou de mobilité.
- NetApp ne recommande pas d'utiliser la croissance automatique FlexVol® dans tous les pilotes ONTAP, sauf ontap-san. En guise de solution de contournement, Trident prend en charge l'utilisation de la réserve de snapshots et adapte les volumes FlexVol® en conséquence.

Autorisations de l'utilisateur

Trident doit être exécuté en tant qu'administrateur ONTAP ou SVM, généralement en utilisant l' `admin`utilisateur cluster ou un `vsadmin`utilisateur SVM, ou un utilisateur avec un nom différent ayant le même rôle. Pour les déploiements Amazon FSx for NetApp ONTAP, Trident doit être exécuté en tant qu'administrateur ONTAP ou SVM, en utilisant l'utilisateur cluster `fsxadmin` ou un `vsadmin`utilisateur SVM, ou un utilisateur avec un nom différent ayant le même rôle. L' `fsxadmin`utilisateur est un remplacement limité pour l'utilisateur administrateur du cluster.

REMARQUE	Si vous utilisez le `limitAggregateUsage` paramètre, des autorisations d'administrateur de cluster sont requises. Lors de l'utilisation d'Amazon FSx for NetApp ONTAP avec Trident, le `limitAggregateUsage` paramètre ne fonctionnera pas avec les comptes d'utilisateur `vsadmin` et `fsxadmin`. L'opération de configuration échouera si vous spécifiez ce paramètre.

Bien qu'il soit possible de créer un rôle plus restrictif au sein d'ONTAP qu'un pilote Trident peut utiliser, nous ne le recommandons pas. La plupart des nouvelles versions de Trident feront appel à des API supplémentaires dont il faudrait tenir compte, ce qui rend les mises à niveau difficiles et sujettes aux erreurs.

Considérations supplémentaires pour NVMe/TCP

Trident prend en charge le protocole non-volatile memory express (NVMe) à l'aide du `ontap-san` driver, notamment :

- IPv6
- Instantanés et clones de volumes NVMe
- Redimensionnement d'un volume NVMe
- Importation d'un volume NVMe créé en dehors de Trident afin que son cycle de vie puisse être géré par Trident
- Multipathing natif NVMe
- Arrêt progressif ou brutal des nœuds K8s (24.06)

Trident ne prend pas en charge :

- DH-HMAC-CHAP pris en charge nativement par NVMe
- Multipathing du device mapper (DM)
- chiffrement LUKS

REMARQUE	NVMe est pris en charge uniquement avec les API REST ONTAP et non avec ONTAPI (ZAPI).

Préparez-vous à configurer le backend avec les pilotes SAN ONTAP

Comprenez les exigences et les options d'authentification pour configurer un backend ONTAP avec des pilotes SAN ONTAP.

Exigences

Pour tous les backends ONTAP, Trident exige qu'au moins un agrégat soit affecté au SVM.

REMARQUE	"Systèmes ASA r2" diffèrent des autres systèmes ONTAP (ASA, AFF et FAS) par l'implémentation de leur couche de stockage. Dans les systèmes ASA r2, des zones de disponibilité de stockage sont utilisées au lieu des agrégats. Consultez l'article de la "ce" Knowledge Base pour savoir comment affecter des agrégats aux SVM dans les systèmes ASA r2.

N'oubliez pas que vous pouvez également exécuter plusieurs pilotes et créer des classes de stockage qui pointent vers l'un ou l'autre. Par exemple, vous pouvez configurer une `san-dev` classe qui utilise le `ontap-`

`san` pilote et une `san-default` classe qui utilise le `ontap-san-economy` pilote.

Tous vos nœuds de travail Kubernetes doivent disposer des outils iSCSI appropriés. Consultez "Préparez le nœud de travail" pour plus de détails.

Authentifier le backend ONTAP

Trident propose deux modes d'authentification pour un backend ONTAP.

- Authentification par identifiants : le nom d'utilisateur et le mot de passe d'un utilisateur ONTAP disposant des autorisations requises. Il est recommandé d'utiliser un rôle de connexion de sécurité prédéfini, tel que `admin` ou `vsadmin` pour garantir une compatibilité maximale avec les versions d'ONTAP.
- Authentification par certificat : Trident peut également communiquer avec un cluster ONTAP à l'aide d'un certificat installé sur le backend. Dans ce cas, la définition du backend doit contenir les valeurs encodées en Base64 du certificat client, de la clé et, si utilisé (recommandé), du certificat de l'autorité de certification de confiance.

Vous pouvez mettre à jour les backends existants pour basculer entre les méthodes basées sur les identifiants et celles basées sur les certificats. Cependant, une seule méthode d'authentification est prise en charge à la fois. Pour passer à une autre méthode d'authentification, vous devez supprimer la méthode existante de la configuration du backend.

AVERTISSEMENT	Si vous tentez de fournir **à la fois des informations d'identification et des certificats**, la création du backend échouera avec une erreur indiquant que plus d'une méthode d'authentification a été fournie dans le fichier de configuration.

Activer l'authentification basée sur les identifiants

Trident requiert les identifiants d'un administrateur au niveau SVM ou au niveau cluster pour communiquer avec le backend ONTAP. Il est recommandé d'utiliser des rôles standard prédéfinis tels que `admin` ou `vsadmin`. Cela garantit la compatibilité avec les futures versions d'ONTAP qui pourraient exposer des API de fonctionnalités à utiliser par les futures versions de Trident. Un rôle de connexion de sécurité personnalisé peut être créé et utilisé avec Trident, mais cela n'est pas recommandé.

Voici un exemple de définition de backend :

YAML

```yaml
---
version: 1
backendName: ExampleBackend
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_nfs
username: vsadmin
password: password
```

JSON

```json
{
  "version": 1,
  "backendName": "ExampleBackend",
  "storageDriverName": "ontap-san",
  "managementLIF": "10.0.0.1",
  "svm": "svm_nfs",
  "username": "vsadmin",
  "password": "password"
}
```

Il est important de noter que la définition du backend est le seul endroit où les identifiants sont stockés en clair. Après la création du backend, les noms d'utilisateur et mots de passe sont encodés avec Base64 et stockés comme secrets Kubernetes. La création ou la mise à jour d'un backend est la seule étape qui nécessite la connaissance des identifiants. En tant que telle, il s'agit d'une opération réservée à l'administrateur Kubernetes/stockage.

Activer l'authentification basée sur les certificats

Les nouveaux et les anciens backends peuvent utiliser un certificat et communiquer avec le backend ONTAP. Trois paramètres sont requis dans la définition du backend.

- clientCertificate: Valeur codée en Base64 du certificat client.
- clientPrivateKey : valeur encodée en Base64 de la clé privée associée.
- trustedCACertificate : valeur encodée en Base64 du certificat CA de confiance. Si vous utilisez un CA de confiance, ce paramètre doit être fourni. Cela peut être ignoré si aucun CA de confiance n'est utilisé.

Un workflow typique implique les étapes suivantes.

Étapes

1. Générez un certificat client et une clé. Lors de la génération, définissez le nom commun (CN) sur l'utilisateur ONTAP à authentifier.

```
openssl req -x509 -nodes -days 1095 -newkey rsa:2048 -keyout k8senv.key
-out k8senv.pem -subj "/C=US/ST=NC/L=RTP/O=NetApp/CN=admin"
```

2. Ajoutez un certificat d'autorité de certification de confiance au cluster ONTAP. Cette opération peut déjà être effectuée par l'administrateur de stockage. Ignorez si aucune autorité de certification de confiance n'est utilisée.

```
security certificate install -type server -cert-name <trusted-ca-cert-
name> -vserver <vserver-name>
ssl modify -vserver <vserver-name> -server-enabled true -client-enabled
true -common-name <common-name> -serial <SN-from-trusted-CA-cert> -ca
<cert-authority>
```

3. Installez le certificat client et la clé (de l'étape 1) sur le cluster ONTAP.

```
security certificate install -type client-ca -cert-name <certificate-
name> -vserver <vserver-name>
security ssl modify -vserver <vserver-name> -client-enabled true
```

REMARQUE	Après avoir exécuté cette commande, ONTAP vous invite à saisir un certificat. Collez le contenu du `k8senv.pem` fichier généré à l'étape 1, puis appuyez sur `END` pour terminer l'installation.

4. Confirmez que le rôle de connexion de sécurité ONTAP prend en charge `cert` la méthode d'authentification.

```
security login create -user-or-group-name admin -application ontapi
-authentication-method cert
security login create -user-or-group-name admin -application http
-authentication-method cert
```

5. Testez l'authentification à l'aide du certificat généré. Remplacez <ONTAP Management LIF> et <vserver name> par l'adresse IP de la LIF de gestion et le nom SVM.

```
curl -X POST -Lk https://<ONTAP-Management-
LIF>/servlets/netapp.servlets.admin.XMLrequest_filer --key k8senv.key
--cert ~/k8senv.pem -d '<?xml version="1.0" encoding="UTF-8"?><netapp
xmlns="http://www.netapp.com/filer/admin" version="1.21"
vfiler="<vserver-name>"><vserver-get></vserver-get></netapp>'
```

6. Encodez le certificat, la clé et le certificat d'autorité de certification de confiance avec Base64.

```
base64 -w 0 k8senv.pem >> cert_base64
base64 -w 0 k8senv.key >> key_base64
base64 -w 0 trustedca.pem >> trustedca_base64
```

7. Créez le backend en utilisant les valeurs obtenues à l'étape précédente.

```
cat cert-backend.json
{
"version": 1,
"storageDriverName": "ontap-san",
"backendName": "SanBackend",
"managementLIF": "1.2.3.4",
"svm": "vserver_test",
"clientCertificate": "Faaaakkkkeeee...Vaaalllluuuueeee",
"clientPrivateKey": "LS0tFaKE...0VaLuES0tLS0K",
"trustedCACertificate": "QNFinfO...SiqOyN",
"storagePrefix": "myPrefix_"
}

tridentctl create backend -f cert-backend.json -n trident
+------------+----------------+--------------------------------------
+--------+---------+
|    NAME    | STORAGE DRIVER |                 UUID                 |
STATE  | VOLUMES |
+------------+----------------+--------------------------------------
+--------+---------+
| SanBackend | ontap-san      | 586b1cd5-8cf8-428d-a76c-2872713612c1 |
online |       0 |
+------------+----------------+--------------------------------------
+--------+---------+
```

Mettez à jour les méthodes d'authentification ou faites pivoter les identifiants

Vous pouvez mettre à jour un backend existant pour utiliser une méthode d'authentification différente ou pour faire pivoter ses identifiants. Cela fonctionne dans les deux sens : les backends qui utilisent un nom d'utilisateur/mot de passe peuvent être mis à jour pour utiliser des certificats ; les backends qui utilisent des certificats peuvent être mis à jour pour utiliser un nom d'utilisateur/mot de passe. Pour ce faire, vous devez supprimer la méthode d'authentification existante et ajouter la nouvelle méthode d'authentification. Ensuite, utilisez le fichier backend.json mis à jour contenant les paramètres requis pour exécuter `tridentctl backend update`.

```
cat cert-backend-updated.json
{
"version": 1,
"storageDriverName": "ontap-san",
"backendName": "SanBackend",
"managementLIF": "1.2.3.4",
"svm": "vserver_test",
"username": "vsadmin",
"password": "password",
"storagePrefix": "myPrefix_"
}

#Update backend with tridentctl
tridentctl update backend SanBackend -f cert-backend-updated.json -n
trident
+------------+----------------+--------------------------------------
+--------+---------+
|    NAME    | STORAGE DRIVER |                 UUID                 |
STATE  | VOLUMES |
+------------+----------------+--------------------------------------
+--------+---------+
| SanBackend | ontap-san      | 586b1cd5-8cf8-428d-a76c-2872713612c1 |
online |       9 |
+------------+----------------+--------------------------------------
+--------+---------+
```

REMARQUE	Lors du renouvellement des mots de passe, l'administrateur du stockage doit d'abord mettre à jour le mot de passe de l'utilisateur sur ONTAP. Cette opération est suivie d'une mise à jour du backend. Lors du renouvellement des certificats, plusieurs certificats peuvent être associés à l'utilisateur. Le backend est ensuite mis à jour pour utiliser le nouveau certificat, après quoi l'ancien certificat peut être supprimé du cluster ONTAP.

La mise à jour d'un backend n'interrompt pas l'accès aux volumes déjà créés et n'affecte pas les connexions de volumes établies ultérieurement. Une mise à jour réussie du backend indique que Trident peut communiquer avec le backend ONTAP et gérer les futures opérations sur les volumes.

Créer un rôle ONTAP personnalisé pour Trident

Vous pouvez créer un rôle de cluster ONTAP avec des privilèges minimaux afin de ne pas avoir à utiliser le rôle d'administrateur ONTAP pour effectuer des opérations dans Trident. Lorsque vous incluez le nom d'utilisateur dans une configuration backend Trident, Trident utilise le rôle de cluster ONTAP que vous avez créé pour effectuer les opérations.

Reportez-vous à "Générateur de rôles personnalisés Trident" pour plus d'informations sur la création de rôles personnalisés Trident.

Utilisation de l'interface de ligne de commande ONTAP

1. Créez un nouveau rôle à l'aide de la commande suivante :

```
security login role create <role_name\> -cmddirname "command" -access all
–vserver <svm_name\>
```

2. Créez un nom d'utilisateur pour l'utilisateur Trident :

```
security login create -username <user_name\> -application ontapi
-authmethod <password\> -role <name_of_role_in_step_1\> –vserver
<svm_name\> -comment "user_description"
```

3. Associez le rôle à l'utilisateur :

```
security login modify username <user_name\> –vserver <svm_name\> -role
<role_name\> -application ontapi -application console -authmethod
<password\>
```

Utilisation de System Manager

Effectuez les étapes suivantes dans ONTAP System Manager :

1. **Créer un rôle personnalisé**:
   a. Pour créer un rôle personnalisé au niveau du cluster, sélectionnez **Cluster > Settings**.

      (Ou) Pour créer un rôle personnalisé au niveau de la SVM, sélectionnez **Storage > Storage VMs > `required SVM`> Paramètres > Utilisateurs et rôles**.

   b. Sélectionnez l'icône flèche (→) à côté de **Users and Roles**.
   c. Sélectionnez **+Add** sous **Roles**.
   d. Définissez les règles du rôle et cliquez sur **Save**.
2. **Associez le rôle à l'utilisateur Trident** : + Effectuez les étapes suivantes sur la page **Utilisateurs et rôles** :
   a. Sélectionnez l'icône Ajouter **+** sous **Users**.
   b. Sélectionnez le nom d'utilisateur requis, puis sélectionnez un rôle dans le menu déroulant pour **Rôle**.
   c. Cliquez sur **Enregistrer**.

Reportez-vous aux pages suivantes pour plus d'informations :

- "Rôles personnalisés pour l'administration d'ONTAP" ou "Définir des rôles personnalisés"
- "Travailler avec les rôles et les utilisateurs"

Authentifiez les connexions avec CHAP bidirectionnel

Trident peut authentifier les sessions iSCSI avec CHAP bidirectionnel pour les `ontap-san` et `ontap-san-economy` pilotes. Cela nécessite l'activation de l'option `useCHAP` dans la définition de votre backend. Lorsqu'elle est définie sur `true`, Trident configure la sécurité par défaut de l'initiateur du SVM sur CHAP bidirectionnel et définit le nom d'utilisateur et les secrets à partir du fichier backend. NetApp recommande

d'utiliser CHAP bidirectionnel pour authentifier les connexions. Voir l'exemple de configuration suivant :

```
---
version: 1
storageDriverName: ontap-san
backendName: ontap_san_chap
managementLIF: 192.168.0.135
svm: ontap_iscsi_svm
useCHAP: true
username: vsadmin
password: password
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSd6cNwxyz
```

AVERTISSEMENT	Le `useCHAP` paramètre est une option booléenne qui ne peut être configurée qu'une seule fois. Il est défini sur faux par défaut. Après l'avoir défini sur vrai, vous ne pouvez plus le définir sur faux.

En plus de `useCHAP=true`, les `chapInitiatorSecret`, `chapTargetInitiatorSecret`, `chapTargetUsername` et `chapUsername`, les champs doivent être inclus dans la définition du backend. Les secrets peuvent être modifiés après la création d'un backend en exécutant `tridentctl update`.

Comment ça marche

En définissant `useCHAP` sur true, l'administrateur de stockage demande à Trident de configurer CHAP sur le système de stockage. Cela inclut les éléments suivants :

- Configuration de CHAP sur la SVM :
  - Si le type de sécurité par défaut de l'initiateur SVM est « aucun » (défini par défaut) **et** qu'il n'y a pas de LUN préexistants déjà présents dans le volume, Trident définira le type de sécurité par défaut à `CHAP` et procédera à la configuration du nom d'utilisateur et des secrets de l'initiateur CHAP et de la cible.
  - Si la SVM contient des LUN, Trident n'activera pas CHAP sur la SVM. Cela garantit que l'accès aux LUN déjà présentes sur la SVM n'est pas restreint.
- Configuration du nom d'utilisateur et des secrets de l'initiateur et de la cible CHAP ; ces options doivent être spécifiées dans la configuration du backend (comme indiqué ci-dessus).

Après la création du backend, Trident crée une CRD correspondante `tridentbackend` et stocke les secrets CHAP ainsi que les noms d'utilisateur en tant que secrets Kubernetes. Tous les PV créés par Trident sur ce backend seront montés et attachés via CHAP.

Renouvelez les identifiants et mettez à jour les backends

Vous pouvez mettre à jour les informations d'identification CHAP en modifiant les paramètres CHAP dans le `backend.json` fichier. Cela nécessitera la mise à jour des secrets CHAP et l'utilisation de la `tridentctl update` commande pour appliquer ces modifications.

AVERTISSEMENT	Lors de la mise à jour des secrets CHAP pour un backend, vous devez utiliser `tridentctl` pour mettre à jour le backend. Ne mettez pas à jour les informations d'identification sur le cluster de stockage à l'aide de l'ONTAP CLI ou d'ONTAP System Manager, car Trident ne pourra pas prendre en compte ces modifications.

```
cat backend-san.json
{
    "version": 1,
    "storageDriverName": "ontap-san",
    "backendName": "ontap_san_chap",
    "managementLIF": "192.168.0.135",
    "svm": "ontap_iscsi_svm",
    "useCHAP": true,
    "username": "vsadmin",
    "password": "password",
    "chapInitiatorSecret": "cl9qxUpDaTeD",
    "chapTargetInitiatorSecret": "rqxigXgkeUpDaTeD",
    "chapTargetUsername": "iJF4heBRT0TCwxyz",
    "chapUsername": "uh2aNCLSd6cNwxyz",
}

./tridentctl update backend ontap_san_chap -f backend-san.json -n trident
+----------------+----------------+--------------------------------------
+--------+---------+
|   NAME         | STORAGE DRIVER |                 UUID                 |
STATE   | VOLUMES |
+----------------+----------------+--------------------------------------
+--------+---------+
| ontap_san_chap | ontap-san      | aa458f3b-ad2d-4378-8a33-1a472ffbeb5c |
online |       7 |
+----------------+----------------+--------------------------------------
+--------+---------+
```

Les connexions existantes restent inchangées ; elles demeureront actives si les identifiants sont mis à jour par Trident sur le SVM. Les nouvelles connexions utilisent les identifiants mis à jour et les connexions existantes demeurent actives. La déconnexion puis la reconnexion d'anciens PV entraînera leur utilisation des identifiants mis à jour.

Options et exemples de configuration SAN ONTAP

Découvrez comment créer et utiliser des pilotes SAN ONTAP avec votre installation Trident. Cette section fournit des exemples de configuration backend et des détails pour le mappage des backends à StorageClasses. "Systèmes ASA r2" diffèrent des autres systèmes ONTAP (ASA, AFF et FAS) par l'implémentation de leur couche de stockage. Ces variations ont un impact sur l'utilisation de certains paramètres comme noté. "En

savoir plus sur les différences entre les systèmes ASA r2 et les autres systèmes ONTAP". Dans la configuration du backend Trident, il n'est pas nécessaire de préciser que votre système est ASA r2. Lorsque vous sélectionnez `ontap-san` comme `storageDriverName`, Trident détecte automatiquement les systèmes ASA r2 ou autres systèmes ONTAP. Certains paramètres de configuration du backend ne s'appliquent pas aux systèmes ASA r2, comme indiqué dans le tableau ci-dessous.

REMARQUE	Seul le `ontap-san`pilote (avec les protocoles iSCSI, NVMe/TCP et FC) est pris en charge pour les systèmes ASA r2.

Options de configuration du backend

Consultez le tableau suivant pour les options de configuration du backend :

Paramètre	Description	Défaut
`version`		Toujours 1
`storageDriverName`	Nom du pilote de stockage	`ontap-san` ou `ontap-san-economy`
`backendName`	Nom personnalisé ou le stockage backend	Nom du pilote + "_" + dataLIF
`managementLIF`	Adresse IP d'une LIF de gestion de cluster ou de SVM. Un nom de domaine complet (FQDN) peut être spécifié. Il est possible de configurer l'utilisation d'adresses IPv6 si Trident a été installé avec l'option IPv6. Les adresses IPv6 doivent être définies entre crochets, comme `[28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]`. Pour une transition en douceur MetroCluster, consultez le Exemple MetroCluster. **REMARQUE** : Si vous utilisez les identifiants « vsadmin », `managementLIF` doit être celui de la SVM ; si vous utilisez les identifiants « admin », `managementLIF` doit être celui du cluster.	"10.0.0.1", "[2001:1234:abcd::fefe]"

Paramètre	Description	Défaut
`dataLIF`	Adresse IP de l'interface logique de protocole (LIF). Il est possible de configurer l'utilisation d'adresses IPv6 si Trident a été installé avec l'option IPv6. Les adresses IPv6 doivent être définies entre crochets, comme `[28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]`. **Ne pas spécifier pour iSCSI.** Trident utilise "Mappage LUN sélectif ONTAP" pour découvrir les LIF iSCSI nécessaires à l'établissement d'une session multi-chemin. Un avertissement est généré si `dataLIF` est explicitement défini. **Omettre pour MetroCluster.** Voir le Exemple MetroCluster.	Dérivé par le SVM
`svm`	Machine virtuelle de stockage à utiliser **Omettre pour MetroCluster.** Voir le Exemple MetroCluster.	Dérivé si un SVM `managementLIF` est spécifié
`useCHAP`	Utilisez CHAP pour authentifier iSCSI pour les pilotes SAN ONTAP [paramètre booléen]. Définissez sur `true` pour que Trident configure et utilise CHAP bidirectionnel comme authentification par défaut pour la SVM indiquée dans le backend. Consultez "Préparez-vous à configurer le backend avec les pilotes SAN ONTAP" pour plus de détails. **Non pris en charge pour FCP ou NVMe/TCP.**	`false`
`chapInitiatorSecret`	Secret de l'initiateur CHAP. Obligatoire si `useCHAP=true`	""
`labels`	Ensemble d'étiquettes arbitraires au format JSON à appliquer aux volumes	""
`chapTargetInitiatorSecret`	Secret de l'initiateur de la cible CHAP. Obligatoire si `useCHAP=true`	""
`chapUsername`	Nom d'utilisateur entrant. Obligatoire si `useCHAP=true`	""
`chapTargetUsername`	Nom d'utilisateur cible. Obligatoire si `useCHAP=true`	""
`clientCertificate`	Valeur encodée en Base64 du certificat client. Utilisé pour l'authentification par certificat	""
`clientPrivateKey`	Valeur encodée en Base64 de la clé privée du client. Utilisé pour l'authentification par certificat	""
`trustedCACertificate`	Valeur encodée en Base64 du certificat d'autorité de certification de confiance. Facultatif. Utilisé pour l'authentification par certificat.	""
`username`	Nom d'utilisateur requis pour communiquer avec le cluster ONTAP. Utilisé pour l'authentification par identifiants. Pour l'authentification Active Directory, voir "Authentifier Trident auprès d'un SVM backend à l'aide des identifiants Active Directory".	""

Paramètre	Description	Défaut
`password`	Mot de passe requis pour communiquer avec le cluster ONTAP. Utilisé pour l'authentification par identifiants. Pour l'authentification Active Directory, voir "Authentifier Trident auprès d'un SVM backend à l'aide des identifiants Active Directory".	""
`svm`	Machine virtuelle de stockage à utiliser	Dérivé si un SVM `managementLIF` est spécifié
`storagePrefix`	Préfixe utilisé lors du provisionnement de nouveaux volumes dans la SVM. Ne peut pas être modifié ultérieurement. Pour mettre à jour ce paramètre, vous devrez créer un nouveau backend.	`trident`
`aggregate`	Agrégat pour le provisionnement (facultatif ; s'il est défini, il doit être attribué à la SVM). Pour le `ontap-nas-flexgroup` driver, cette option est ignorée. S'il n'est pas attribué, n'importe quel agrégat disponible peut être utilisé pour provisionner un FlexGroup volume. **REMARQUE** : Lorsqu'un agrégat est mis à jour dans SVM, il est automatiquement mis à jour dans Trident par interrogation de SVM, sans qu'il soit nécessaire de redémarrer le contrôleur Trident. Si vous avez configuré un agrégat spécifique dans Trident pour provisionner des volumes, si l'agrégat est renommé ou déplacé hors de la SVM, le backend passera en état d'échec dans Trident lors de l'interrogation de l'agrégat SVM. Vous devez soit changer l'agrégat pour un qui est présent sur la SVM, soit le supprimer complètement pour remettre le backend en ligne. **Ne pas spécifier pour les systèmes ASA r2**.	""
`limitAggregateUsage`	L'approvisionnement échoue si l'utilisation dépasse ce pourcentage. Si vous utilisez un Amazon FSx for NetApp ONTAP backend, ne spécifiez pas `limitAggregateUsage`. Les `fsxadmin` et `vsadmin` fournis ne contiennent pas les autorisations requises pour récupérer l'utilisation agrégée et la limiter à l'aide de Trident. **Ne pas spécifier pour les systèmes ASA r2**.	"" (non appliqué par défaut)

Paramètre	Description	Défaut
`limitVolumeSize`	L'approvisionnement échoue si la taille du volume demandée dépasse cette valeur. Limite également la taille maximale des volumes qu'il gère pour les LUN.	"" (non appliqué par défaut)
`lunsPerFlexvol`	Nombre maximal de LUN par FlexVol®, doit être compris dans la plage [50, 200]	`100`
`debugTraceFlags`	Options de débogage à utiliser lors du dépannage. Exemple, {"api":false, "method":true} n'utilisez-les que si vous effectuez un dépannage et avez besoin d'un journal détaillé.	`null`
`useREST`	Paramètre booléen à utiliser pour les API REST ONTAP. `` `useREST` Lorsqu'il est défini sur `true`, Trident utilise les API REST ONTAP pour communiquer avec le backend ; lorsqu'il est défini sur `false`, Trident utilise les appels ONTAPI (ZAPI) pour communiquer avec le backend. Cette fonctionnalité nécessite ONTAP 9.11.1 ou une version ultérieure. De plus, le rôle de connexion ONTAP utilisé doit avoir accès à l'application `ontapi`. Cela est satisfait par les rôles prédéfinis `vsadmin` et `cluster-admin`. À partir de la version Trident 24.06 et ONTAP 9.15.1 ou version ultérieure, `useREST` est défini sur `true` par défaut ; modifiez `useREST` sur `false` pour utiliser les appels ONTAPI (ZAPI). `` `useREST` est entièrement compatible NVMe/TCP. **REMARQUE** : NVMe est pris en charge uniquement avec les API REST ONTAP et non avec ONTAPI (ZAPI). **Si spécifié, toujours définir sur `true` pour les systèmes ASA r2**.	`true` pour ONTAP 9.15.1 ou version ultérieure, sinon `false`.

Paramètre	Description	Défaut
`sanType`	Utilisez pour sélectionner `iscsi` pour iSCSI, `nvme` pour NVMe/TCP ou `fcp` pour SCSI sur Fibre Channel (FC).	`iscsi` si vide
`formatOptions`	Utilisez `formatOptions` pour spécifier des arguments de ligne de commandes pour la commande `mkfs`, qui seront appliqués chaque fois qu'un volume est formaté. Cela vous permet de formater le volume selon vos préférences. Assurez-vous de spécifier les formatOptions similaires à celles des options de la commande mkfs, à l'exclusion du chemin du périphérique. Exemple : « -E nodiscard » **Pris en charge pour `ontap-san` et `ontap-san-economy` pilotes avec le protocole iSCSI. En outre, pris en charge pour les systèmes ASA r2 lors de l'utilisation des protocoles iSCSI et NVMe/TCP.**	
`limitVolumePoolSize`	Taille maximale requise de FlexVol lors de l'utilisation de LUN dans le backend ontap-san-economy.	"" (non appliqué par défaut)
`denyNewVolumePools`	Limite `ontap-san-economy` les backends à la création de nouveaux volumes FlexVol® pour contenir leurs LUN. Seuls les FlexVol® préexistants sont utilisés pour le provisionnement de nouveaux PV.	

Recommandations d'utilisation de formatOptions

Trident recommande les options suivantes pour accélérer le processus de mise en forme :

- **-E nodiscard (ext3, ext4) :** Ne pas tenter de supprimer des blocs lors de la création du système de fichiers (la suppression initiale de blocs est utile sur les disques SSD et les systèmes de stockage à provisionnement fin). Cette option remplace l'option obsolète « -K » et s'applique aux systèmes de fichiers ext3 et ext4.
- **-K (xfs) :** Ne pas tenter de supprimer des blocs lors de la création du système de fichiers (mkfs). Cette option s'applique au système de fichiers xfs.

Authentifier Trident auprès d'un SVM backend à l'aide des identifiants Active Directory

Vous pouvez configurer Trident pour qu'il s'authentifie auprès d'une SVM backend à l'aide d'identifiants Active Directory (AD). Avant qu'un compte AD puisse accéder à la SVM, vous devez configurer l'accès des contrôleurs de domaine AD au cluster ou à la SVM. Pour l'administration du cluster avec un compte AD, vous devez créer un tunnel de domaine. Consultez "Configurer l'accès au contrôleur de domaine Active Directory dans ONTAP" pour plus de détails.

étapes

1. Configurer les paramètres du système de noms de domaine (DNS) pour une SVM backend :

```
vserver services dns create -vserver <svm_name> -dns-servers
<dns_server_ip1>,<dns_server_ip2>
```

2. Exécutez la commande suivante pour créer un compte d'ordinateur pour la SVM dans Active Directory :

```
vserver active-directory create -vserver DataSVM -account-name ADSERVER1
-domain demo.netapp.com
```

3. Utilisez cette commande pour créer un utilisateur ou un groupe AD afin de gérer le cluster ou la SVM

```
security login create -vserver <svm_name> -user-or-group-name
<ad_user_or_group> -application <application> -authentication-method domain
-role vsadmin
```

4. Dans le fichier de configuration backend de Trident, définissez les paramètres `username` et `password` sur le nom d'utilisateur ou de groupe AD et le mot de passe, respectivement.

Options de configuration backend pour le provisionnement des volumes

Vous pouvez contrôler le provisionnement par défaut à l'aide de ces options dans la section `defaults` de la configuration. Pour un exemple, consultez les exemples de configuration ci-dessous.

Paramètre	Description	Défaut
`spaceAllocation`	Allocation d'espace pour les LUNs	"true" **Si spécifié, définissez sur `true` pour les systèmes ASA r2**.
`spaceReserve`	Mode de réservation d'espace : « aucun » (fin) ou « volume » (épais). **À définir sur `none` pour les systèmes ASA r2**.	"none"
`snapshotPolicy`	Stratégie d'instantané à utiliser. **À définir sur `none` pour les systèmes ASA r2**.	"none"
`qosPolicy`	Groupe de règles QoS à attribuer aux volumes créés. Choisissez l'un de qosPolicy ou adaptiveQosPolicy par pool de stockage/backend. L'utilisation des groupes de règles QoS avec Trident requiert ONTAP 9.8 ou une version ultérieure. Vous devez utiliser un groupe de règles QoS non partagé et vous assurer que le groupe de règles est appliqué individuellement à chaque composant. Un groupe de règles QoS partagé impose une limite au débit total de toutes les charges de travail.	""
`adaptiveQosPolicy`	Groupe de règles QoS adaptatives à attribuer aux volumes créés. Choisissez l'un des deux, qosPolicy ou adaptiveQosPolicy, par pool de stockage/backend	""
`snapshotReserve`	Pourcentage du volume réservé aux snapshots. **Ne pas spécifier pour les systèmes ASA r2**.	"0" si `snapshotPolicy` est "aucun", sinon ""
`splitOnClone`	Séparer un clone de son parent lors de sa création	"false"

Paramètre	Description	Défaut
`encryption`	Activez le chiffrement de volume NetApp (NVE) sur le nouveau volume ; la valeur par défaut est `false`. NVE doit être sous licence et activé sur le cluster pour utiliser cette option. Si NAE est activé sur le backend, tout volume provisionné dans Trident sera activé pour NAE. Pour plus d'informations, consultez : "Comment Trident fonctionne avec NVE et NAE".	"false" **Si spécifié, définissez sur `true` pour les systèmes ASA r2**.
`luksEncryption`	Activez le chiffrement LUKS. Consultez "Utilisez Linux Unified Key Setup (LUKS)".	"" **Définir sur `false` pour les systèmes ASA r2**.
`tieringPolicy`	Politique de hiérarchisation à utiliser « none » **Ne pas spécifier pour les systèmes ASA r2**.	
`nameTemplate`	Modèle pour créer des noms de volumes personnalisés.	""

Exemples de provisionnement de volumes

Voici un exemple avec des valeurs par défaut définies :

```
---
version: 1
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: trident_svm
username: admin
password: <password>
labels:
  k8scluster: dev2
  backend: dev2-sanbackend
storagePrefix: alternate-trident
debugTraceFlags:
  api: false
  method: true
defaults:
  spaceReserve: volume
  qosPolicy: standard
  spaceAllocation: 'false'
  snapshotPolicy: default
  snapshotReserve: '10'
```

REMARQUE	Pour tous les volumes créés à l'aide du `ontap-san` driver, Trident ajoute 10 % de capacité supplémentaire à la FlexVol pour intégrer les métadonnées LUN. La LUN sera provisionnée avec la taille exacte demandée par l'utilisateur dans le PVC. Trident ajoute 10 % à la FlexVol (affiché comme taille disponible dans ONTAP). Les utilisateurs obtiendront désormais la capacité utilisable qu'ils ont demandée. Cette modification empêche également les LUN de devenir en lecture seule, sauf si l'espace disponible est entièrement utilisé. Ceci ne s'applique pas à ontap-san-economy.

Pour les backends qui définissent `snapshotReserve`, Trident calcule la taille des volumes comme suit :

```
Total volume size = [(PVC requested size) / (1 - (snapshotReserve
percentage) / 100)] * 1.1
```

Le chiffre 1,1 correspond aux 10 % supplémentaires que Trident ajoute au FlexVol pour prendre en compte les métadonnées LUN. Pour `snapshotReserve` = 5 %, et une demande PVC = 5 Gio, la taille totale du volume est de 5,79 Gio et la taille disponible est de 5,5 Gio. La commande `volume show` devrait afficher des résultats similaires à cet exemple :

Actuellement, le redimensionnement est le seul moyen d'utiliser le nouveau calcul pour un volume existant.

Exemples de configuration minimale

Les exemples suivants présentent des configurations de base qui laissent la plupart des paramètres par défaut. C'est la manière la plus simple de définir un backend.

REMARQUE	Si vous utilisez Amazon FSx sur NetApp ONTAP avec Trident, NetApp recommande de spécifier les noms DNS des LIF au lieu des adresses IP.

Exemple SAN ONTAP

Il s'agit d'une configuration de base utilisant le `ontap-san` driver.

```
---
version: 1
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_iscsi
labels:
  k8scluster: test-cluster-1
  backend: testcluster1-sanbackend
username: vsadmin
password: <password>
```

Exemple MetroCluster

Vous pouvez configurer le backend pour éviter d'avoir à mettre à jour manuellement la définition du backend après le basculement et le retour en arrière pendant "Réplication et récupération de SVM".

Pour une transition et un retour en arrière sans heurt, spécifiez le SVM en utilisant `managementLIF` et omettez les `svm` paramètres. Par exemple :

```
version: 1
storageDriverName: ontap-san
managementLIF: 192.168.1.66
username: vsadmin
password: password
```

Exemple d'économie SAN ONTAP

```
version: 1
storageDriverName: ontap-san-economy
managementLIF: 10.0.0.1
svm: svm_iscsi_eco
username: vsadmin
password: <password>
```

Exemple d'authentification par certificat

Dans cet exemple de configuration de base `clientCertificate`, `clientPrivateKey`, et `trustedCACertificate` (facultatif, si vous utilisez une autorité de certification de confiance) sont renseignés dans `backend.json` et prennent respectivement les valeurs encodées en base64 du certificat client, de la clé privée et du certificat de l'autorité de certification de confiance.

```
---
version: 1
storageDriverName: ontap-san
backendName: DefaultSANBackend
managementLIF: 10.0.0.1
svm: svm_iscsi
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSd6cNwxyz
clientCertificate: ZXR0ZXJwYXB...ICMgJ3BhcGVyc2
clientPrivateKey: vciwKIyAgZG...0cnksIGRlc2NyaX
trustedCACertificate: zcyBbaG...b3Igb3duIGNsYXNz
```

Exemples CHAP bidirectionnels

Ces exemples créent un backend avec `useCHAP` défini sur `true`.

Exemple ONTAP SAN CHAP

```
---
version: 1
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_iscsi
labels:
  k8scluster: test-cluster-1
  backend: testcluster1-sanbackend
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSd6cNwxyz
username: vsadmin
password: <password>
```

Exemple d'économie ONTAP SAN CHAP

```
---
version: 1
storageDriverName: ontap-san-economy
managementLIF: 10.0.0.1
svm: svm_iscsi_eco
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSd6cNwxyz
username: vsadmin
password: <password>
```

Exemple NVMe/TCP

Vous devez disposer d'une SVM configurée avec NVMe sur votre backend ONTAP. Il s'agit d'une configuration de base pour NVMe/TCP.

```
---
version: 1
backendName: NVMeBackend
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_nvme
username: vsadmin
password: password
sanType: nvme
useREST: true
```

Exemple de SCSI sur FC (FCP)

Vous devez disposer d'une SVM configurée avec FC sur votre backend ONTAP. Il s'agit d'une configuration de base du backend pour FC.

```
---
version: 1
backendName: fcp-backend
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_fc
username: vsadmin
password: password
sanType: fcp
useREST: true
```

Exemple de configuration du backend avec nameTemplate

```
---
version: 1
storageDriverName: ontap-san
backendName: ontap-san-backend
managementLIF: <ip address>
svm: svm0
username: <admin>
password: <password>
defaults:
  nameTemplate:
"{{.volume.Name}}_{{.labels.cluster}}_{{.volume.Namespace}}_{{.vo\
    lume.RequestName}}"
labels:
  cluster: ClusterA
  PVC: "{{.volume.Namespace}}_{{.volume.RequestName}}"
```

formatOptions exemple pour le pilote ontap-san-economy

```
---
version: 1
storageDriverName: ontap-san-economy
managementLIF: ""
svm: svm1
username: ""
password: "!"
storagePrefix: whelk_
debugTraceFlags:
  method: true
  api: true
defaults:
  formatOptions: -E nodiscard
```

Exemples de backends avec pools virtuels

Dans ces exemples de fichiers de définition de backend, des valeurs par défaut spécifiques sont définies pour tous les pools de stockage, comme `spaceReserve` à aucun, `spaceAllocation` à faux, et `encryption` à faux. Les pools virtuels sont définis dans la section storage.

Trident définit les étiquettes de provisionnement dans le champ « Commentaires ». Les commentaires sont définis sur le volume FlexVol. Trident copie toutes les étiquettes présentes sur un pool virtuel vers le volume de stockage lors du provisionnement. Pour plus de simplicité, les administrateurs de stockage peuvent définir des étiquettes par pool virtuel et regrouper les volumes par étiquette.

Dans ces exemples, certains pools de stockage définissent leurs propres `spaceReserve`, `spaceAllocation`, et `encryption` valeurs, et certains pools remplacent les valeurs par défaut.

Exemple SAN ONTAP

```yaml
---
version: 1
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_iscsi
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSd6cNwxyz
username: vsadmin
password: <password>
defaults:
  spaceAllocation: "false"
  encryption: "false"
  qosPolicy: standard
labels:
  store: san_store
  kubernetes-cluster: prod-cluster-1
region: us_east_1
storage:
  - labels:
      protection: gold
      creditpoints: "40000"
    zone: us_east_1a
    defaults:
      spaceAllocation: "true"
      encryption: "true"
      adaptiveQosPolicy: adaptive-extreme
  - labels:
      protection: silver
      creditpoints: "20000"
    zone: us_east_1b
    defaults:
      spaceAllocation: "false"
      encryption: "true"
      qosPolicy: premium
  - labels:
      protection: bronze
      creditpoints: "5000"
    zone: us_east_1c
    defaults:
      spaceAllocation: "true"
      encryption: "false"
```

Exemple d'économie SAN ONTAP

```
---
version: 1
storageDriverName: ontap-san-economy
managementLIF: 10.0.0.1
svm: svm_iscsi_eco
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSd6cNwxyz
username: vsadmin
password: <password>
defaults:
  spaceAllocation: "false"
  encryption: "false"
labels:
  store: san_economy_store
region: us_east_1
storage:
  - labels:
      app: oracledb
      cost: "30"
    zone: us_east_1a
    defaults:
      spaceAllocation: "true"
      encryption: "true"
  - labels:
      app: postgresdb
      cost: "20"
    zone: us_east_1b
    defaults:
      spaceAllocation: "false"
      encryption: "true"
  - labels:
      app: mysqldb
      cost: "10"
    zone: us_east_1c
    defaults:
      spaceAllocation: "true"
      encryption: "false"
  - labels:
      department: legal
      creditpoints: "5000"
    zone: us_east_1c
```

```
    defaults:
      spaceAllocation: "true"
      encryption: "false"
```

Exemple NVMe/TCP

```
---
version: 1
storageDriverName: ontap-san
sanType: nvme
managementLIF: 10.0.0.1
svm: nvme_svm
username: vsadmin
password: <password>
useREST: true
defaults:
  spaceAllocation: "false"
  encryption: "true"
storage:
  - labels:
      app: testApp
      cost: "20"
    defaults:
      spaceAllocation: "false"
      encryption: "false"
```

Map backends vers StorageClasses

Les définitions suivantes de StorageClass se réfèrent à Exemples de backends avec pools virtuels. En utilisant le champ `parameters.selector`, chaque StorageClass indique quels pools virtuels peuvent être utilisés pour héberger un volume. Le volume possédera les aspects définis dans le pool virtuel choisi.

- Le `protection-gold` StorageClass sera associé au premier pool virtuel dans le `ontap-san` backend. Il s'agit du seul pool offrant une protection de niveau or.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-gold
provisioner: csi.trident.netapp.io
parameters:
  selector: "protection=gold"
  fsType: "ext4"
```

- Le `protection-not-gold` StorageClass sera associé aux deuxième et troisième pools virtuels dans `ontap-san` backend. Ce sont les seuls pools offrant un niveau de protection autre que gold.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-not-gold
provisioner: csi.trident.netapp.io
parameters:
  selector: "protection!=gold"
  fsType: "ext4"
```

- Le `app-mysqldb` StorageClass correspondra au troisième pool virtuel dans le `ontap-san-economy` backend. Il s'agit du seul pool offrant une configuration de pool de stockage pour l'app de type mysqldb.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: app-mysqldb
provisioner: csi.trident.netapp.io
parameters:
  selector: "app=mysqldb"
  fsType: "ext4"
```

- Le `protection-silver-creditpoints-20k` StorageClass sera associé au deuxième pool virtuel dans `ontap-san` backend. Il s'agit du seul pool offrant une protection de niveau argent et 20000 points de crédit.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-silver-creditpoints-20k
provisioner: csi.trident.netapp.io
parameters:
  selector: "protection=silver; creditpoints=20000"
  fsType: "ext4"
```

- Le `creditpoints-5k` StorageClass sera associé au troisième pool virtuel dans le `ontap-san` backend et au quatrième pool virtuel dans le `ontap-san-economy` backend. Ce sont les seuls pools proposant 5000 points de crédit.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: creditpoints-5k
provisioner: csi.trident.netapp.io
parameters:
  selector: "creditpoints=5000"
  fsType: "ext4"
```

- Le `my-test-app-sc` StorageClass sera associé au `testAPP` pool virtuel dans le `ontap-san` pilote avec `sanType: nvme`. C'est le seul pool offrant `testApp`.

```
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: my-test-app-sc
provisioner: csi.trident.netapp.io
parameters:
  selector: "app=testApp"
  fsType: "ext4"
```

Trident déterminera quel pool virtuel est sélectionné et s'assurera que l'exigence de stockage est respectée.

Pilotes ONTAP NAS

Présentation du pilote ONTAP NAS

Découvrez comment configurer un backend ONTAP avec les pilotes NAS ONTAP et Cloud Volumes ONTAP.

Détails du pilote ONTAP NAS

Trident fournit les pilotes de stockage NAS suivants pour communiquer avec le cluster ONTAP. Les modes d'accès pris en charge sont : *ReadWriteOnce* (RWO), *ReadOnlyMany* (ROX), *ReadWriteMany* (RWX), *ReadWriteOncePod* (RWOP).

Pilote	Protocole	volumeMode	Modes d'accès pris en charge	Systèmes de fichiers pris en charge
`ontap-nas`	NFS SMB	Système de fichiers	RWO, ROX, RWX, RWOP	"", `nfs`, `smb`
`ontap-nas-economy`	NFS SMB	Système de fichiers	RWO, ROX, RWX, RWOP	"", `nfs`, `smb`

Pilote	Protocole	volumeMode	Modes d'accès pris en charge	Systèmes de fichiers pris en charge
`ontap-nas-flexgroup`	NFS SMB	Système de fichiers	RWO, ROX, RWX, RWOP	"", `nfs`, `smb`

AVERTISSEMENT

- Utilisez `ontap-san-economy` uniquement si le nombre d'utilisations du volume persistant doit être supérieur à "limites de volume ONTAP prises en charge".
- Utilisez `ontap-nas-economy` uniquement si le nombre d'utilisations de volumes persistants est censé être supérieur à "limites de volume ONTAP prises en charge" et que le pilote `ontap-san-economy` ne peut pas être utilisé.
- N'utilisez pas `ontap-nas-economy` si vous prévoyez un besoin de protection des données, de reprise après sinistre ou de mobilité.
- NetApp ne recommande pas d'utiliser la croissance automatique FlexVol® dans tous les pilotes ONTAP, sauf ontap-san. En guise de solution de contournement, Trident prend en charge l'utilisation de la réserve de snapshots et adapte les volumes FlexVol® en conséquence.

Autorisations de l'utilisateur

Trident doit être exécuté en tant qu'administrateur ONTAP ou SVM, généralement en utilisant l' `admin`utilisateur cluster ou un `vsadmin`utilisateur SVM, ou un utilisateur avec un nom différent ayant le même rôle.

Pour les déploiements Amazon FSx for NetApp ONTAP, Trident doit être exécuté en tant qu'administrateur ONTAP ou SVM, en utilisant l'utilisateur cluster `fsxadmin` ou un `vsadmin`utilisateur SVM, ou un utilisateur avec un nom différent ayant le même rôle. L' `fsxadmin`utilisateur est un remplacement limité pour l'utilisateur administrateur du cluster.

REMARQUE

Si vous utilisez le `limitAggregateUsage` paramètre, des autorisations d'administrateur de cluster sont requises. Lors de l'utilisation d'Amazon FSx for NetApp ONTAP avec Trident, le `limitAggregateUsage` paramètre ne fonctionnera pas avec les comptes d'utilisateur `vsadmin` et `fsxadmin`. L'opération de configuration échouera si vous spécifiez ce paramètre.

Bien qu'il soit possible de créer un rôle plus restrictif au sein d'ONTAP qu'un pilote Trident peut utiliser, nous ne le recommandons pas. La plupart des nouvelles versions de Trident feront appel à des API supplémentaires dont il faudrait tenir compte, ce qui rend les mises à niveau difficiles et sujettes aux erreurs.

Préparez-vous à configurer un backend avec les pilotes NAS ONTAP

Comprenez les exigences, les options d'authentification et les règles d'export pour configurer un backend ONTAP avec les pilotes ONTAP NAS. À compter de la version 25.10, NetApp Trident prend en charge "NetApp système de stockage AFX". Les systèmes de stockage NetApp AFX diffèrent des autres systèmes ONTAP (ASA, AFF et FAS) dans l'implémentation de leur couche de stockage. Dans la configuration du backend Trident, il n'est pas nécessaire de préciser que votre système est AFX. Lorsque

vous sélectionnez `ontap-nas` comme `storageDriverName`, Trident détecte automatiquement les systèmes AFX.

REMARQUE	Seul le `ontap-nas` driver (avec le protocole NFS) est pris en charge pour les systèmes AFX; le protocole SMB n'est pas pris en charge.

Exigences

- Pour tous les backends ONTAP, Trident exige qu'au moins un agrégat soit affecté au SVM.
- Vous pouvez exécuter plusieurs pilotes et créer des classes de stockage qui pointent vers l'un ou l'autre. Par exemple, vous pouvez configurer une classe Gold qui utilise le `ontap-nas` driver et une classe Bronze qui utilise le `ontap-nas-economy` driver.
- Tous vos nœuds de travail Kubernetes doivent disposer des outils NFS appropriés. Consultez "ici" pour plus de détails.
- Trident prend uniquement en charge les volumes SMB montés sur des pods exécutés sur des nœuds Windows. Consultez Préparez-vous à provisionner des volumes SMB pour plus de détails.

Authentifier le backend ONTAP

Trident propose deux modes d'authentification pour un backend ONTAP.

- Basé sur les identifiants : ce mode requiert des autorisations suffisantes sur le backend ONTAP. Il est recommandé d'utiliser un compte associé à un rôle de connexion de sécurité prédéfini, tel que `admin` ou `vsadmin` afin de garantir une compatibilité maximale avec les versions ONTAP.
- Authentification par certificat : ce mode requiert l'installation d'un certificat sur le backend pour que Trident puisse communiquer avec un ONTAP cluster. Dans ce cas, la définition du backend doit contenir les valeurs encodées en Base64 du certificat client, de la clé et, si utilisé (recommandé), du certificat de l'autorité de certification de confiance.

Vous pouvez mettre à jour les backends existants pour basculer entre les méthodes basées sur les identifiants et celles basées sur les certificats. Cependant, une seule méthode d'authentification est prise en charge à la fois. Pour passer à une autre méthode d'authentification, vous devez supprimer la méthode existante de la configuration du backend.

AVERTISSEMENT	Si vous tentez de fournir **à la fois des informations d'identification et des certificats**, la création du backend échouera avec une erreur indiquant que plus d'une méthode d'authentification a été fournie dans le fichier de configuration.

Activer l'authentification basée sur les identifiants

Trident requiert les identifiants d'un administrateur au niveau SVM ou au niveau cluster pour communiquer avec le backend ONTAP. Il est recommandé d'utiliser des rôles standard prédéfinis tels que `admin` ou `vsadmin`. Cela garantit la compatibilité avec les futures versions d'ONTAP qui pourraient exposer des API de fonctionnalités à utiliser par les futures versions de Trident. Un rôle de connexion de sécurité personnalisé peut être créé et utilisé avec Trident, mais cela n'est pas recommandé.

Voici un exemple de définition de backend :

YAML

```yaml
---
version: 1
backendName: ExampleBackend
storageDriverName: ontap-nas
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
svm: svm_nfs
credentials:
  name: secret-backend-creds
```

JSON

```json
{
  "version": 1,
  "backendName": "ExampleBackend",
  "storageDriverName": "ontap-nas",
  "managementLIF": "10.0.0.1",
  "dataLIF": "10.0.0.2",
  "svm": "svm_nfs",
  "credentials": {
        "name": "secret-backend-creds"
    }
}
```

Il est important de noter que la définition du backend est le seul endroit où les identifiants sont stockés en clair. Après la création du backend, les noms d'utilisateur et mots de passe sont encodés avec Base64 et stockés comme secrets Kubernetes. La création/la mise à jour d'un backend est la seule étape nécessitant la connaissance des identifiants. En tant que telle, il s'agit d'une opération réservée à l'administrateur Kubernetes/stockage.

Activer l'authentification par certificat

Les nouveaux et les anciens backends peuvent utiliser un certificat et communiquer avec le backend ONTAP. Trois paramètres sont requis dans la définition du backend.

- clientCertificate: Valeur codée en Base64 du certificat client.
- clientPrivateKey : valeur encodée en Base64 de la clé privée associée.
- trustedCACertificate : valeur encodée en Base64 du certificat CA de confiance. Si vous utilisez un CA de confiance, ce paramètre doit être fourni. Cela peut être ignoré si aucun CA de confiance n'est utilisé.

Un workflow typique implique les étapes suivantes.

Étapes

1. Générez un certificat client et une clé. Lors de la génération, définissez le nom commun (CN) sur

l'utilisateur ONTAP à authentifier.

```
openssl req -x509 -nodes -days 1095 -newkey rsa:2048 -keyout k8senv.key
-out k8senv.pem -subj "/C=US/ST=NC/L=RTP/O=NetApp/CN=vsadmin"
```

2. Ajoutez un certificat d'autorité de certification de confiance au cluster ONTAP. Cette opération peut déjà être effectuée par l'administrateur de stockage. Ignorez si aucune autorité de certification de confiance n'est utilisée.

```
security certificate install -type server -cert-name <trusted-ca-cert-
name> -vserver <vserver-name>
ssl modify -vserver <vserver-name> -server-enabled true -client-enabled
true -common-name <common-name> -serial <SN-from-trusted-CA-cert> -ca
<cert-authority>
```

3. Installez le certificat client et la clé (de l'étape 1) sur le cluster ONTAP.

```
security certificate install -type client-ca -cert-name <certificate-
name> -vserver <vserver-name>
security ssl modify -vserver <vserver-name> -client-enabled true
```

4. Confirmez que le rôle de connexion de sécurité ONTAP prend en charge `cert` la méthode d'authentification.

```
security login create -user-or-group-name vsadmin -application ontapi
-authentication-method cert -vserver <vserver-name>
security login create -user-or-group-name vsadmin -application http
-authentication-method cert -vserver <vserver-name>
```

5. Testez l'authentification à l'aide du certificat généré. Remplacez <ONTAP Management LIF> et <vserver name> par l'adresse IP de la LIF de gestion et le nom SVM. Vous devez vous assurer que le LIF a sa stratégie de service définie sur `default-data-management`.

```
curl -X POST -Lk https://<ONTAP-Management-
LIF>/servlets/netapp.servlets.admin.XMLrequest_filer --key k8senv.key
--cert ~/k8senv.pem -d '<?xml version="1.0" encoding="UTF-8"?><netapp
xmlns="http://www.netapp.com/filer/admin" version="1.21"
vfiler="<vserver-name>"><vserver-get></vserver-get></netapp>'
```

6. Encodez le certificat, la clé et le certificat d'autorité de certification de confiance avec Base64.

```
base64 -w 0 k8senv.pem >> cert_base64
base64 -w 0 k8senv.key >> key_base64
base64 -w 0 trustedca.pem >> trustedca_base64
```

7. Créez le backend en utilisant les valeurs obtenues à l'étape précédente.

```
cat cert-backend-updated.json
{
"version": 1,
"storageDriverName": "ontap-nas",
"backendName": "NasBackend",
"managementLIF": "1.2.3.4",
"dataLIF": "1.2.3.8",
"svm": "vserver_test",
"clientCertificate": "Faaaakkkkeeee...Vaaalllluuuueeee",
"clientPrivateKey": "LS0tFaKE...0VaLuES0tLS0K",
"storagePrefix": "myPrefix_"
}

#Update backend with tridentctl
tridentctl update backend NasBackend -f cert-backend-updated.json -n
trident
+------------+----------------+--------------------------------------
+--------+---------+
|    NAME    | STORAGE DRIVER |                 UUID                 |
STATE  | VOLUMES |
+------------+----------------+--------------------------------------
+--------+---------+
| NasBackend | ontap-nas      | 98e19b74-aec7-4a3d-8dcf-128e5033b214 |
online |       9 |
+------------+----------------+--------------------------------------
+--------+---------+
```

Mettez à jour les méthodes d'authentification ou faites pivoter les identifiants

Vous pouvez mettre à jour un backend existant pour utiliser une méthode d'authentification différente ou pour faire pivoter ses identifiants. Cela fonctionne dans les deux sens : les backends qui utilisent un nom d'utilisateur/mot de passe peuvent être mis à jour pour utiliser des certificats ; les backends qui utilisent des certificats peuvent être mis à jour pour utiliser un nom d'utilisateur/mot de passe. Pour ce faire, vous devez supprimer la méthode d'authentification existante et ajouter la nouvelle méthode d'authentification. Ensuite, utilisez le fichier backend.json mis à jour contenant les paramètres requis pour exécuter `tridentctl update backend`.

```
cat cert-backend-updated.json
```

```
{
"version": 1,
"storageDriverName": "ontap-nas",
"backendName": "NasBackend",
"managementLIF": "1.2.3.4",
"dataLIF": "1.2.3.8",
"svm": "vserver_test",
"username": "vsadmin",
"password": "password",
"storagePrefix": "myPrefix_"
}
```

```
#Update backend with tridentctl
tridentctl update backend NasBackend -f cert-backend-updated.json -n
trident
+------------+----------------+--------------------------------------
+--------+---------+
|    NAME    | STORAGE DRIVER |                 UUID                 |
STATE  | VOLUMES |
+------------+----------------+--------------------------------------
+--------+---------+
| NasBackend | ontap-nas      | 98e19b74-aec7-4a3d-8dcf-128e5033b214 |
online |       9 |
+------------+----------------+--------------------------------------
+--------+---------+
```

REMARQUE	Lors du renouvellement des mots de passe, l'administrateur du stockage doit d'abord mettre à jour le mot de passe de l'utilisateur sur ONTAP. Cette opération est suivie d'une mise à jour du backend. Lors du renouvellement des certificats, plusieurs certificats peuvent être associés à l'utilisateur. Le backend est ensuite mis à jour pour utiliser le nouveau certificat, après quoi l'ancien certificat peut être supprimé du cluster ONTAP.

La mise à jour d'un backend n'interrompt pas l'accès aux volumes déjà créés et n'affecte pas les connexions de volumes établies ultérieurement. Une mise à jour réussie du backend indique que Trident peut communiquer avec le backend ONTAP et gérer les futures opérations sur les volumes.

Créer un rôle ONTAP personnalisé pour Trident

Vous pouvez créer un rôle de cluster ONTAP avec des privilèges minimaux afin de ne pas avoir à utiliser le rôle d'administrateur ONTAP pour effectuer des opérations dans Trident. Lorsque vous incluez le nom d'utilisateur dans une configuration backend Trident, Trident utilise le rôle de cluster ONTAP que vous avez

créé pour effectuer les opérations.

Reportez-vous à "Générateur de rôles personnalisés Trident" pour plus d'informations sur la création de rôles personnalisés Trident.

Utilisation de l'interface de ligne de commande ONTAP

1. Créez un nouveau rôle à l'aide de la commande suivante :

```
security login role create <role_name\> -cmddirname "command" -access all
–vserver <svm_name\>
```

2. Créez un nom d'utilisateur pour l'utilisateur Trident :

```
security login create -username <user_name\> -application ontapi
-authmethod <password\> -role <name_of_role_in_step_1\> –vserver
<svm_name\> -comment "user_description"
```

3. Associez le rôle à l'utilisateur :

```
security login modify username <user_name\> –vserver <svm_name\> -role
<role_name\> -application ontapi -application console -authmethod
<password\>
```

Utilisation de System Manager

Effectuez les étapes suivantes dans ONTAP System Manager :

1. **Créer un rôle personnalisé**:
   a. Pour créer un rôle personnalisé au niveau du cluster, sélectionnez **Cluster > Settings**.

      (Ou) Pour créer un rôle personnalisé au niveau de la SVM, sélectionnez **Storage > Storage VMs > `required SVM`> Paramètres > Utilisateurs et rôles**.

   b. Sélectionnez l'icône flèche (→) à côté de **Users and Roles**.
   c. Sélectionnez **+Add** sous **Roles**.
   d. Définissez les règles du rôle et cliquez sur **Save**.
2. **Associez le rôle à l'utilisateur Trident** : + Effectuez les étapes suivantes sur la page **Utilisateurs et rôles** :
   a. Sélectionnez l'icône Ajouter **+** sous **Users**.
   b. Sélectionnez le nom d'utilisateur requis, puis sélectionnez un rôle dans le menu déroulant pour **Rôle**.
   c. Cliquez sur **Enregistrer**.

Reportez-vous aux pages suivantes pour plus d'informations :

- "Rôles personnalisés pour l'administration d'ONTAP" ou "Définir des rôles personnalisés"
- "Travailler avec les rôles et les utilisateurs"

Gérer les règles d'export NFS

Trident utilise des règles d'export NFS pour contrôler l'accès aux volumes qu'il provisionne.

Trident propose deux options lors de l'utilisation des règles d'export :

- Trident peut gérer dynamiquement la règle d'export ; dans ce mode de fonctionnement, l'administrateur de stockage spécifie une liste de blocs CIDR représentant les adresses IP admissibles. Trident ajoute automatiquement à la règle d'export les adresses IP des nœuds concernés qui appartiennent à ces plages lors de la publication. Sinon, si aucun CIDR n'est spécifié, toutes les adresses IP unicast à portée globale trouvées sur le nœud auquel le volume est publié seront ajoutées à la règle d'export.
- Les administrateurs de stockage peuvent créer une règle d'export et y ajouter des règles manuellement. Trident utilise la règle d'export par défaut, sauf si un autre nom de règle d'export est spécifié dans la configuration.

Gérer dynamiquement les règles d'export

Trident offre la possibilité de gérer dynamiquement les règles d'export pour les backends ONTAP. Cela donne à l'administrateur de stockage la possibilité de spécifier un espace d'adresses autorisé pour les adresses IP des nœuds de travail, plutôt que de définir manuellement des règles explicites. Cela simplifie grandement la gestion des règles d'export ; les modifications apportées à la règle d'export ne nécessitent plus d'intervention manuelle sur le cluster de stockage. De plus, cela permet de restreindre l'accès au cluster de stockage uniquement aux nœuds de travail qui montent des volumes et dont les adresses IP se trouvent dans la plage spécifiée, ce qui permet une gestion fine et automatisée.

REMARQUE	N'utilisez pas la traduction d'adresses réseau (NAT) lorsque vous utilisez des règles d'export dynamiques. Avec la NAT, le contrôleur de stockage voit l'adresse NAT du frontal et non l'adresse IP réelle de l'hôte, donc l'accès sera refusé si aucune correspondance n'est trouvée dans les règles d'export.

Exemple

Deux options de configuration doivent être utilisées. Voici un exemple de définition de backend :

```
---
version: 1
storageDriverName: ontap-nas-economy
backendName: ontap_nas_auto_export
managementLIF: 192.168.0.135
svm: svm1
username: vsadmin
password: password
autoExportCIDRs:
  - 192.168.0.0/24
autoExportPolicy: true
```

REMARQUE	Lorsque vous utilisez cette fonctionnalité, vous devez vous assurer que la jonction racine de votre SVM possède une règle d'export précédemment créée avec une règle d'export autorisant le bloc CIDR du nœud (par exemple, la règle d'export par défaut). Suivez toujours la bonne pratique recommandée par NetApp pour dédier une SVM à Trident.

Voici une explication du fonctionnement de cette fonctionnalité à l'aide de l'exemple ci-dessus :

- `autoExportPolicy` est défini sur `true`. Cela indique que Trident crée une règle d'export pour chaque volume provisionné avec ce backend pour la `svm1` SVM et gère l'ajout et la suppression de règles à l'aide de blocs d'adresses `autoexportCIDRs`. Jusqu'à ce qu'un volume soit attaché à un nœud, le volume utilise une règle d'export vide sans aucune règle afin d'empêcher tout accès non désiré à ce volume. Lorsqu'un volume est publié sur un nœud, Trident crée une règle d'export portant le même nom que le qtree sous-jacent contenant l'adresse IP du nœud dans le bloc CIDR spécifié. Ces adresses IP seront également ajoutées à la règle d'export utilisée par le volume FlexVol parent.
  - Par exemple :
    - UUID de backend 403b5326-8482-40db-96d0-d83fb3f4daec
    - `autoExportPolicy` défini sur `true`
    - préfixe de stockage `trident`
    - UUID PVC a79bcf5f-7b6d-4a40-9876-e2551f159c1c
    - Le qtree nommé trident_pvc_a79bcf5f_7b6d_4a40_9876_e2551f159c1c crée une règle d'export pour le FlexVol nommé `trident-403b5326-8482-40db96d0-d83fb3f4daec`, une règle d'export pour le qtree nommé `trident_pvc_a79bcf5f_7b6d_4a40_9876_e2551f159c1c`, et une règle d'export vide nommée `trident_empty` sur la SVM. Les règles de la règle d'export du FlexVol seront un sur-ensemble de toutes les règles contenues dans les règles d'export du qtree. La règle d'export vide sera réutilisée par tous les volumes qui ne sont pas attachés.
- `autoExportCIDRs` contient une liste de blocs d'adresses. Ce champ est facultatif et il est défini par défaut sur ["0.0.0.0/0", "::/0"]. S'il n'est pas défini, Trident ajoute toutes les adresses unicast à portée globale trouvées sur les nœuds de travail avec des publications.

Dans cet exemple, l' `192.168.0.0/24`espace d'adresses est fourni. Cela indique que les adresses IP des nœuds Kubernetes qui se trouvent dans cette plage d'adresses avec des publications seront ajoutées à la règles d'export que Trident crée. Lorsque Trident enregistre un nœud sur lequel il s'exécute, il récupère les adresses IP du nœud et les vérifie par rapport aux blocs d'adresses fournis dans `autoExportCIDRs`. Au moment de la publication, après avoir filtré les adresses IP, Trident crée les règles d'export pour les adresses IP clientes du nœud auquel il publie.

Vous pouvez mettre à jour `autoExportPolicy` et `autoExportCIDRs` pour les backends après leur création. Vous pouvez ajouter de nouveaux CIDR pour un backend géré automatiquement ou supprimer les CIDR existants. Faites attention lors de la suppression de CIDR afin de garantir que les connexions existantes ne soient pas interrompues. Vous pouvez également choisir de désactiver `autoExportPolicy` pour un backend et revenir à une règle d'export créée manuellement. Cela nécessitera de définir le paramètre `exportPolicy` dans la configuration de votre backend.

Après que Trident a créé ou mis à jour un backend, vous pouvez vérifier le backend en utilisant `tridentctl` ou le CRD correspondant `tridentbackend` :

```
./tridentctl get backends ontap_nas_auto_export -n trident -o yaml
items:
- backendUUID: 403b5326-8482-40db-96d0-d83fb3f4daec
  config:
    aggregate: ""
    autoExportCIDRs:
    - 192.168.0.0/24
    autoExportPolicy: true
    backendName: ontap_nas_auto_export
    chapInitiatorSecret: ""
    chapTargetInitiatorSecret: ""
    chapTargetUsername: ""
    chapUsername: ""
    dataLIF: 192.168.0.135
    debug: false
    debugTraceFlags: null
    defaults:
      encryption: "false"
      exportPolicy: <automatic>
      fileSystemType: ext4
```

Lorsqu'un nœud est supprimé, Trident vérifie toutes les règles d'export afin de supprimer les règles d'accès correspondantes au nœud. En supprimant cette adresse IP de nœud des règles d'export des backends gérés, Trident empêche les montages non autorisés, sauf si cette adresse IP est réutilisée par un nouveau nœud dans le cluster.

Pour les backends existants, la mise à jour du backend avec `tridentctl update backend` garantit que Trident gère automatiquement les règles d'export. Cela crée deux nouvelles règles d'export nommées d'après l'UUID du backend et le nom du qtree lorsqu'elles sont nécessaires. Les volumes présents sur le backend utiliseront les nouvelles règles d'export après avoir été démontés puis remontés.

REMARQUE	La suppression d'un backend avec des règles d'export gérées automatiquement supprimera la règle d'export créée dynamiquement. Si le backend est recréé, il est traité comme un nouveau backend et cela entraînera la création d'une nouvelle règle d'export.

Si l'adresse IP d'un nœud en production est modifiée, vous devez redémarrer le pod Trident sur ce nœud. Trident mettra alors à jour la règle d'export pour les backends qu'il gère afin de refléter ce changement d'adresse IP.

Préparez-vous à provisionner des volumes SMB

Avec un peu de préparation supplémentaire, vous pouvez provisionner des volumes SMB à l'aide de `ontap-nas` drivers.

AVERTISSEMENT	Vous devez configurer les protocoles NFS et SMB/CIFS sur la SVM pour créer un `ontap-nas-economy` volume SMB pour les clusters ONTAP sur site. L'absence de configuration de l'un ou l'autre de ces protocoles entraînera l'échec de la création du volume SMB.

REMARQUE	`autoExportPolicy` n'est pas pris en charge pour les volumes SMB.

Avant de commencer

Avant de pouvoir provisionner des volumes SMB, vous devez disposer des éléments suivants.

- Un cluster Kubernetes avec un nœud contrôleur Linux et au moins un nœud de travail Windows exécutant Windows Server 2022. Trident prend uniquement en charge les volumes SMB montés sur des pods exécutés sur des nœuds Windows.
- Au moins un secret Trident contenant vos identifiants Active Directory. Pour générer le secret `smbcreds` :

```
kubectl create secret generic smbcreds --from-literal username=user
--from-literal password='password'
```

- Un proxy CSI configuré en tant que service Windows. Pour configurer un `csi-proxy`, reportez-vous à "GitHub : CSI Proxy" ou "GitHub: CSI Proxy pour Windows" pour les nœuds Kubernetes exécutés sous Windows.

Étapes

1. Pour ONTAP sur site, vous pouvez éventuellement créer un partage SMB ou Trident peut en créer un pour vous.

REMARQUE	Les partages SMB sont requis pour Amazon FSx pour ONTAP.

   Vous pouvez créer les partages d'administration SMB de deux manières : soit à l'aide du "Microsoft Management Console" Shared Folders snap-in, soit à l'aide de l'interface de ligne de commande ONTAP. Pour créer les partages SMB à l'aide de l'interface de ligne de commande ONTAP :

   a. Si nécessaire, créez la structure du chemin d'accès du répertoire pour le partage.

      La `vserver cifs share create`commande vérifie le chemin spécifié dans l'option -path lors de la création du partage. Si le chemin spécifié n'existe pas, la commande échoue.

   b. Créer un partage SMB associé à la SVM spécifiée :

      ```
      vserver cifs share create -vserver vserver_name -share-name
      share_name -path path [-share-properties share_properties,...]
      [other_attributes] [-comment text]
      ```

   c. Vérifiez que le partage a été créé :

      ```
      vserver cifs share show -share-name share_name
      ```

REMARQUE Reportez-vous à "Créer un partage SMB" pour plus de détails.

2. Lors de la création du backend, vous devez configurer les éléments suivants pour spécifier les volumes SMB. Pour toutes les options de configuration du backend FSx pour ONTAP, reportez-vous à "Options de configuration et exemples pour FSx for ONTAP".

Paramètre	Description	Exemple
`smbShare`	Vous pouvez spécifier l'une des options suivantes : le nom d'un partage SMB créé à l'aide de la console de gestion Microsoft ou de l'interface de ligne de commande ONTAP ; un nom pour permettre à Trident de créer le partage SMB ; ou vous pouvez laisser le paramètre vide pour empêcher l'accès partagé aux volumes. Ce paramètre est facultatif pour ONTAP sur site. Ce paramètre est obligatoire pour Amazon FSx for ONTAP backends et ne peut pas être vide.	`smb-share`
`nasType`	**Doit être défini sur `smb`.** Si nul, la valeur par défaut est `nfs`.	`smb`
`securityStyle`	Style de sécurité pour les nouveaux volumes. **Doit être défini sur `ntfs` ou `mixed` pour les volumes SMB.**	`ntfs` or `mixed` pour les volumes SMB
`unixPermissions`	Mode pour les nouveaux volumes. **Doit rester vide pour les volumes SMB.**	""

Activer le SMB sécurisé

À partir de la version 25.06, NetApp Trident prend en charge le provisionnement sécurisé des volumes SMB créés à l'aide de `ontap-nas` et `ontap-nas-economy` serveurs backend. Lorsque le protocole SMB sécurisé est activé, vous pouvez fournir un accès contrôlé aux partages SMB pour les utilisateurs et groupes d'utilisateurs Active Directory (AD) à l'aide des listes de contrôle d'accès (ACL).

Points à retenir

- L'importation `ontap-nas-economy` de volumes n'est pas prise en charge.
- Seuls les clones en lecture seule sont pris en charge pour `ontap-nas-economy` volumes.
- Si le protocole SMB sécurisé est activé, Trident ignorera le partage SMB mentionné dans le backend.
- La mise à jour de l'annotation PVC, de l'annotation de classe de stockage et du champ backend ne met pas à jour l'ACL du partage SMB.
- La liste de contrôle d'accès (ACL) de partage SMB spécifiée dans l'annotation du PVC cloné aura priorité sur celles du PVC source.
- Veillez à fournir des utilisateurs AD valides lors de l'activation du protocole SMB sécurisé. Les utilisateurs non valides ne seront pas ajoutés à l'ACL.
- Si vous fournissez le même utilisateur AD dans le backend, la storage class et le PVC avec des autorisations différentes, la priorité des autorisations sera : PVC, storage class, puis backend.
- Le protocole SMB sécurisé est pris en charge pour `ontap-nas` les importations de volumes gérés et n'est pas applicable aux importations de volumes non gérés.

Étapes

1. Spécifiez adAdminUser dans TridentBackendConfig comme indiqué dans l'exemple suivant :

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-nas
  managementLIF: 10.193.176.x
  svm: svm0
  useREST: true
  defaults:
    adAdminUser: tridentADtest
  credentials:
    name: backend-tbc-ontap-invest-secret
```

2. Ajoutez l'annotation dans la classe de stockage.

   Ajoutez l' `trident.netapp.io/smbShareAdUser`annotation à la classe de stockage pour activer SMB sécurisé sans faute. La valeur utilisateur spécifiée pour l'annotation `trident.netapp.io/smbShareAdUser` doit être identique au nom d'utilisateur indiqué dans le `smbcreds` secret. Vous pouvez choisir l'une des options suivantes pour `smbShareAdUserPermission` : `full_control`, `change`, ou `read`. L'autorisation par défaut est `full_control`.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-smb-sc
  annotations:
    trident.netapp.io/smbShareAdUserPermission: change
    trident.netapp.io/smbShareAdUser: tridentADuser
parameters:
  backendType: ontap-nas
  csi.storage.k8s.io/node-stage-secret-name: smbcreds
  csi.storage.k8s.io/node-stage-secret-namespace: trident
  trident.netapp.io/nasType: smb
provisioner: csi.trident.netapp.io
reclaimPolicy: Delete
volumeBindingMode: Immediate
```

1. Créer un PVC.

L'exemple suivant crée un PVC :

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-pvc4
  namespace: trident
  annotations:
    trident.netapp.io/snapshotDirectory: "true"
    trident.netapp.io/smbShareAccessControl: |
      read:
        - tridentADtest
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: ontap-smb-sc
```

Options et exemples de configuration NAS ONTAP

Apprenez à créer et à utiliser des pilotes NAS ONTAP avec votre installation Trident. Cette section fournit des exemples de configuration backend et des détails pour le mappage des backends à StorageClasses. À compter de la version 25.10, NetApp Trident prend en charge "NetApp AFX systèmes de stockage". NetApp AFX les systèmes de stockage diffèrent des autres systèmes basés sur ONTAP (ASA, AFF et FAS) dans l'implémentation de leur couche de stockage.

REMARQUE	Seul le `ontap-nas` driver (avec le protocole NFS) est pris en charge pour les systèmes NetApp AFX ; le protocole SMB n'est pas pris en charge.

Options de configuration du backend

Dans la configuration du backend Trident, il n'est pas nécessaire de préciser que votre système est un NetApp AFX storage system. Lorsque vous sélectionnez `ontap-nas` comme `storageDriverName`, Trident détecte automatiquement le système de stockage AFX. Certains paramètres de configuration du backend ne sont pas applicables aux systèmes de stockage AFX.

Le tableau suivant présente les options de configuration du backend :

Paramètre	Description	Défaut
`version`		Toujours 1

Paramètre	Description	Défaut
`storageDriverName`	Nom du pilote de stockage **REMARQUE** Pour les systèmes NetApp AFX, seul `ontap-nas` est pris en charge.	`ontap-nas`, `ontap-nas-economy`, ou `ontap-nas-flexgroup`
`backendName`	Nom personnalisé ou le stockage backend	Nom du pilote + "_" + dataLIF
`managementLIF`	Adresse IP d'un cluster ou d'une LIF de gestion SVM. Un nom de domaine complet (FQDN) peut être spécifié. Il est possible de configurer l'utilisation d'adresses IPv6 si Trident a été installé avec l'option IPv6. Les adresses IPv6 doivent être définies entre crochets, comme `[28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]`. Pour une transition en douceur MetroCluster, consultez le Exemple MetroCluster.	"10.0.0.1", "[2001:1234:abcd::fefe]"
`dataLIF`	Adresse IP de l'interface logique de protocole (LIF). NetApp recommande de spécifier `dataLIF`. Si elle n'est pas fournie, Trident récupère les dataLIF depuis la SVM. Vous pouvez spécifier un nom de domaine complet (FQDN) à utiliser pour les opérations de montage NFS, ce qui vous permet de créer un DNS à répartition de charge (round-robin) entre plusieurs dataLIF. Peut être modifié après la configuration initiale. Consultez . Il est possible de configurer l'utilisation d'adresses IPv6 si Trident a été installé avec l'option IPv6. Les adresses IPv6 doivent être définies entre crochets, comme `[28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]`. **Omettre pour MetroCluster.** Voir le Exemple MetroCluster.	Adresse spécifiée ou dérivée de la SVM, si non spécifiée (non recommandé)
`svm`	Machine virtuelle de stockage à utiliser **Omettre pour MetroCluster.** Voir le Exemple MetroCluster.	Dérivé si un SVM `managementLIF` est spécifié
`autoExportPolicy`	Activer la création et la mise à jour automatiques des règles d'export [Booléen]. En utilisant les options `autoExportPolicy` et `autoExportCIDRs`, Trident peut gérer les règles d'export automatiquement.	false
`autoExportCIDRs`	Liste des CIDR pour filtrer les adresses IP des nœuds Kubernetes lorsque `autoExportPolicy` est activé. En utilisant les options `autoExportPolicy` et `autoExportCIDRs`, Trident peut gérer les règles d'export automatiquement.	["0.0.0.0/0", "::/0"]`
`labels`	Ensemble d'étiquettes arbitraires au format JSON à appliquer aux volumes	""
`clientCertificate`	Valeur encodée en Base64 du certificat client. Utilisé pour l'authentification par certificat	""

Paramètre	Description	Défaut
`clientPrivateKey`	Valeur encodée en Base64 de la clé privée du client. Utilisé pour l'authentification par certificat	""
`trustedCACertificate`	Valeur encodée en Base64 du certificat d'autorité de certification de confiance. Facultatif. Utilisé pour l'authentification par certificat	""
`username`	Nom d'utilisateur pour se connecter au cluster/SVM. Utilisé pour l'authentification par identifiants. Pour l'authentification Active Directory, voir "Authentifier Trident auprès d'un SVM backend à l'aide des identifiants Active Directory".	
`password`	Mot de passe pour se connecter au cluster/SVM. Utilisé pour l'authentification par identifiants. Pour l'authentification Active Directory, voir "Authentifier Trident auprès d'un SVM backend à l'aide des identifiants Active Directory".	
`storagePrefix`	Préfixe utilisé lors du provisionnement de nouveaux volumes dans la SVM. Ne peut pas être modifié après sa définition **REMARQUE** Lors de l'utilisation d'ontap-nas-economy et d'un storagePrefix de 24 caractères ou plus, les qtrees n'auront pas le préfixe de stockage intégré, bien qu'il soit présent dans le nom du volume.	"Trident"

Paramètre	Description	Défaut
`aggregate`	Agrégat pour le provisionnement (facultatif ; s'il est défini, il doit être attribué à la SVM). Pour le `ontap-nas-flexgroup` driver, cette option est ignorée. S'il n'est pas attribué, n'importe quel agrégat disponible peut être utilisé pour provisionner un FlexGroup volume. **REMARQUE** : Lorsqu'un agrégat est mis à jour dans SVM, il est automatiquement mis à jour dans Trident par interrogation de SVM, sans qu'il soit nécessaire de redémarrer le contrôleur Trident. Si vous avez configuré un agrégat spécifique dans Trident pour provisionner des volumes, si l'agrégat est renommé ou déplacé hors de la SVM, le backend passera en état d'échec dans Trident lors de l'interrogation de l'agrégat SVM. Vous devez soit changer l'agrégat pour un qui est présent sur la SVM, soit le supprimer complètement pour remettre le backend en ligne. **Ne pas spécifier pour les systèmes de stockage AFX.**	""
`limitAggregateUsage`	L'approvisionnement échoue si l'utilisation dépasse ce pourcentage. **Ne s'applique pas à Amazon FSx pour ONTAP**. **Ne pas spécifier pour les systèmes de stockage AFX.**	"" (non appliqué par défaut)

Paramètre	Description	Défaut
flexgroupAggregateList	Liste des agrégats à provisionner (facultatif ; si défini, doit être affecté au SVM). Tous les agrégats affectés au SVM sont utilisés pour provisionner un FlexGroup volume. Pris en charge par le pilote de stockage **ontap-nas-flexgroup**. **REMARQUE** : Lorsque la liste d'agrégats est mise à jour dans SVM, la liste est mise à jour automatiquement dans Trident par interrogation de SVM, sans qu'il soit nécessaire de redémarrer le contrôleur Trident. Lorsque vous avez configuré une liste d'agrégats spécifique dans Trident pour le provisionnement de volumes, si la liste d'agrégats est renommée ou déplacée hors de SVM, le backend passera à l'état d'échec dans Trident lors de l'interrogation de l'agrégat SVM. Vous devez soit modifier la liste d'agrégats pour en choisir une présente sur le SVM, soit la supprimer complètement afin de remettre le backend en ligne.	""
`limitVolumeSize`	L'approvisionnement échoue si la taille du volume demandée dépasse cette valeur.	"" (non appliqué par défaut)
`debugTraceFlags`	Options de débogage à utiliser lors du dépannage. Exemple, {"api":false, "method":true} Ne pas utiliser `debugTraceFlags` sauf si vous effectuez un dépannage et avez besoin d'un journal détaillé.	null
`nasType`	Configurez la création de volumes NFS ou SMB. Les options sont `nfs`, `smb` ou null. La valeur null correspond par défaut à des volumes NFS. **Si spécifié, définissez toujours sur `nfs` pour les systèmes de stockage AFX**.	`nfs`

Paramètre	Description	Défaut
`nfsMountOptions`	Liste d'options de montage NFS séparées par des virgules. Les options de montage pour les volumes persistants Kubernetes sont généralement spécifiées dans les classes de stockage, mais si aucune option de montage n'est spécifiée dans une classe de stockage, Trident utilisera les options de montage spécifiées dans le fichier de configuration du backend de stockage. Si aucune option de montage n'est spécifiée ni dans la classe de stockage ni dans le fichier de configuration, Trident n'appliquera aucune option de montage au volume persistant associé.	""
`qtreesPerFlexvol`	Nombre maximal de Qtrees par FlexVol, doit être compris dans la plage [50, 300]	"200"
`smbShare`	Vous pouvez spécifier l'une des options suivantes : le nom d'un partage SMB créé à l'aide de la console de gestion Microsoft ou de l'interface de ligne de commande ONTAP ; un nom pour permettre à Trident de créer le partage SMB ; ou vous pouvez laisser le paramètre vide pour empêcher l'accès partagé aux volumes. Ce paramètre est facultatif pour ONTAP sur site. Ce paramètre est obligatoire pour Amazon FSx for ONTAP backends et ne peut pas être vide.	`smb-share`
`useREST`	Paramètre booléen à utiliser pour les API REST ONTAP. `useREST` Lorsqu'il est défini sur `true`, Trident utilise les API REST ONTAP pour communiquer avec le backend ; lorsqu'il est défini sur `false`, Trident utilise les appels ONTAPI (ZAPI) pour communiquer avec le backend. Cette fonctionnalité nécessite ONTAP 9.11.1 ou une version ultérieure. De plus, le rôle de connexion ONTAP utilisé doit avoir accès à l'application `ontapi`. Cela est satisfait par les rôles prédéfinis `vsadmin` et `cluster-admin`. À partir de la version Trident 24.06 et ONTAP 9.15.1 ou version ultérieure, `useREST` est défini sur `true` par défaut ; modifiez `useREST` sur `false` pour utiliser les appels ONTAPI (ZAPI). **Si spécifié, définissez toujours sur `true` pour les systèmes de stockage AFX**.	`true` pour ONTAP 9.15.1 ou version ultérieure, sinon `false`.
`limitVolumePoolSize`	Taille maximale requise de FlexVol lors de l'utilisation de Qtrees dans le backend ontap-nas-economy.	"" (non appliqué par défaut)
`denyNewVolumePools`	Empêche `ontap-nas-economy` les serveurs backend de créer de nouveaux volumes FlexVol pour y stocker leurs Qtrees. Seuls les FlexVol® préexistants sont utilisés pour le provisionnement de nouveaux PV.	

Paramètre	Description	Défaut
adAdminUser	Utilisateur ou groupe d'utilisateurs administrateur Active Directory disposant d'un accès complet aux partages SMB. Utilisez ce paramètre pour accorder des droits d'administrateur sur le partage SMB avec un contrôle total.	

Options de configuration backend pour le provisionnement des volumes

Vous pouvez contrôler le provisionnement par défaut à l'aide de ces options dans la section `defaults` de la configuration. Pour un exemple, consultez les exemples de configuration ci-dessous.

Paramètre	Description	Défaut
spaceAllocation	Allocation d'espace pour les Qtrees	"true"
spaceReserve	Mode de réservation d'espace ; "none" (fin) ou "volume" (épais)	"none"
snapshotPolicy	Stratégie de snapshot à utiliser	"none"
qosPolicy	Groupe de règles QoS à attribuer aux volumes créés. Choisissez l'un des deux, qosPolicy ou adaptiveQosPolicy, par pool de stockage/backend	""
adaptiveQosPolicy	Groupe de règles QoS adaptatives à attribuer aux volumes créés. Choisissez l'un de qosPolicy ou adaptiveQosPolicy par pool de stockage/backend. Non pris en charge par ontap-nas-economy.	""
snapshotReserve	Pourcentage du volume réservé aux instantanés	"0" si `snapshotPolicy` est "aucun", sinon ""
splitOnClone	Séparer un clone de son parent lors de sa création	"false"
encryption	Activez le chiffrement de volume NetApp (NVE) sur le nouveau volume ; la valeur par défaut est `false`. NVE doit être sous licence et activé sur le cluster pour utiliser cette option. Si NAE est activé sur le backend, tout volume provisionné dans Trident sera activé pour NAE. Pour plus d'informations, consultez : "Comment Trident fonctionne avec NVE et NAE".	"false"
tieringPolicy	Politique de hiérarchisation à utiliser « none »	
unixPermissions	Mode pour les nouveaux volumes	"777" pour les volumes NFS; vide (non applicable) pour les volumes SMB
snapshotDir	Contrôle l'accès au `.snapshot` répertoire	`true`, `false` (Défini explicitement).
exportPolicy	Règles d'export à utiliser	"default"

Paramètre	Description	Défaut
`securityStyle`	Style de sécurité pour les nouveaux volumes. NFS prend en charge `mixed` et `unix` styles de sécurité. SMB prend en charge `mixed` et `ntfs` styles de sécurité.	La valeur par défaut de NFS est `unix`. La valeur par défaut de SMB est `ntfs`.
`nameTemplate`	Modèle pour créer des noms de volumes personnalisés.	""

REMARQUE	L'utilisation des groupes de règles QoS avec Trident requiert ONTAP 9.8 ou une version ultérieure. Vous devez utiliser un groupe de règles QoS non partagé et vous assurer que le groupe de règles est appliqué individuellement à chaque composant. Un groupe de règles QoS partagé impose une limite au débit total de toutes les charges de travail.

Exemples de provisionnement de volumes

Voici un exemple avec des valeurs par défaut définies :

```
---
version: 1
storageDriverName: ontap-nas
backendName: customBackendName
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
labels:
  k8scluster: dev1
  backend: dev1-nasbackend
svm: trident_svm
username: cluster-admin
password: <password>
limitAggregateUsage: 80%
limitVolumeSize: 50Gi
nfsMountOptions: nfsvers=4
debugTraceFlags:
  api: false
  method: true
defaults:
  spaceReserve: volume
  qosPolicy: premium
  exportPolicy: myk8scluster
  snapshotPolicy: default
  snapshotReserve: "10"
```

Pour `ontap-nas` et `ontap-nas-flexgroups`, Trident utilise désormais un nouveau calcul afin de garantir que le FlexVol soit correctement dimensionné avec le pourcentage de snapshotReserve et le PVC. Lorsque l'utilisateur demande un PVC, Trident crée le FlexVol initial avec plus d'espace en utilisant le nouveau calcul.

Ce calcul garantit que l'utilisateur reçoit l'espace inscriptible qu'il a demandé dans le PVC, et non moins que ce qu'il a demandé. Avant la version v21.07, lorsque l'utilisateur demandait un PVC (par exemple, 5 Gio), avec le snapshotReserve à 50 pour cent, il n'obtenait que 2,5 Gio d'espace inscriptible. Cela s'explique par le fait que ce que l'utilisateur demandait était le volume entier et `snapshotReserve` est un pourcentage de celui-ci. Avec Trident 21.07, ce que l'utilisateur demande est l'espace inscriptible et Trident définit le nombre `snapshotReserve` comme le pourcentage du volume entier. Cela ne s'applique pas à `ontap-nas-economy`. Voir l'exemple suivant pour comprendre comment cela fonctionne :

Le calcul est le suivant :

```
Total volume size = <PVC requested size> / (1 - (<snapshotReserve
percentage> / 100))
```

Pour snapshotReserve = 50 %, et une demande PVC = 5 Gio, la taille totale du volume est de 5 / 0,5 = 10 Gio et la taille disponible est de 5 Gio, ce qui correspond à la demande de l'utilisateur dans la demande PVC. La commande `volume show` devrait afficher des résultats similaires à cet exemple :

Les backends existants issus d'installations précédentes provisionneront les volumes comme expliqué ci-dessus lors de la mise à niveau de Trident. Pour les volumes que vous avez créés avant la mise à niveau, vous devez redimensionner leurs volumes pour que la modification soit prise en compte. Par exemple, un PVC de 2 Gio avec `snapshotReserve=50` aboutissait auparavant à un volume offrant 1 Gio d'espace accessible en écriture. En redimensionnant le volume à 3 Gio, par exemple, l'application disposera de 3 Gio d'espace accessible en écriture sur un volume de 6 Gio.

Exemples de configuration minimale

Les exemples suivants présentent des configurations de base qui laissent la plupart des paramètres par défaut. C'est la manière la plus simple de définir un backend.

REMARQUE	Si vous utilisez Amazon FSx sur NetApp ONTAP avec Trident, la recommandation est de spécifier les noms DNS des LIF au lieu des adresses IP.

Exemple d'économie NAS ONTAP

```
---
version: 1
storageDriverName: ontap-nas-economy
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
svm: svm_nfs
username: vsadmin
password: password
```

Exemple de FlexGroup NAS ONTAP

```
---
version: 1
storageDriverName: ontap-nas-flexgroup
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
svm: svm_nfs
username: vsadmin
password: password
```

Exemple MetroCluster

Vous pouvez configurer le backend pour éviter d'avoir à mettre à jour manuellement la définition du backend après le basculement et le retour en arrière pendant "Réplication et récupération de SVM".

Pour une transition et un retour en arrière sans heurt, spécifiez le SVM en utilisant `managementLIF` et omettez les `dataLIF` et `svm` paramètres. Par exemple :

```
---
version: 1
storageDriverName: ontap-nas
managementLIF: 192.168.1.66
username: vsadmin
password: password
```

Exemple de volumes SMB

```
---
version: 1
backendName: ExampleBackend
storageDriverName: ontap-nas
managementLIF: 10.0.0.1
nasType: smb
securityStyle: ntfs
unixPermissions: ""
dataLIF: 10.0.0.2
svm: svm_nfs
username: vsadmin
password: password
```

Exemple d'authentification par certificat

Voici un exemple de configuration minimale du backend. `clientCertificate`, `clientPrivateKey`, et `trustedCACertificate` (facultatif, si vous utilisez une autorité de certification de confiance) sont renseignés dans `backend.json` et prennent respectivement les valeurs encodées en base64 du certificat client, de la clé privée et du certificat de l'autorité de certification de confiance.

```
---
version: 1
backendName: DefaultNASBackend
storageDriverName: ontap-nas
managementLIF: 10.0.0.1
dataLIF: 10.0.0.15
svm: nfs_svm
clientCertificate: ZXR0ZXJwYXB...ICMgJ3BhcGVyc2
clientPrivateKey: vciwKIyAgZG...0cnksIGRlc2NyaX
trustedCACertificate: zcyBbaG...b3Igb3duIGNsYXNz
storagePrefix: myPrefix_
```

Exemple de règles d'export automatique

Cet exemple vous montre comment vous pouvez demander à Trident d'utiliser des règles d'export dynamiques pour créer et gérer automatiquement la règle d'export. Cela fonctionne de la même manière pour les `ontap-nas-economy` et `ontap-nas-flexgroup` pilotes.

```
---
version: 1
storageDriverName: ontap-nas
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
svm: svm_nfs
labels:
  k8scluster: test-cluster-east-1a
  backend: test1-nasbackend
autoExportPolicy: true
autoExportCIDRs:
- 10.0.0.0/24
username: admin
password: password
nfsMountOptions: nfsvers=4
```

Exemple d'adresses IPv6

Cet exemple montre `managementLIF` l'utilisation d'une adresse IPv6.

```
---
version: 1
storageDriverName: ontap-nas
backendName: nas_ipv6_backend
managementLIF: "[5c5d:5edf:8f:7657:bef8:109b:1b41:d491]"
labels:
  k8scluster: test-cluster-east-1a
  backend: test1-ontap-ipv6
svm: nas_ipv6_svm
username: vsadmin
password: password
```

Exemple d'utilisation d'Amazon FSx for ONTAP avec des volumes SMB

Le `smbShare`paramètre est requis pour FSx for ONTAP utilisant des volumes SMB.

```
---
version: 1
backendName: SMBBackend
storageDriverName: ontap-nas
managementLIF: example.mgmt.fqdn.aws.com
nasType: smb
dataLIF: 10.0.0.15
svm: nfs_svm
smbShare: smb-share
clientCertificate: ZXR0ZXJwYXB...ICMgJ3BhcGVyc2
clientPrivateKey: vciwKIyAgZG...0cnksIGRlc2NyaX
trustedCACertificate: zcyBbaG...b3Igb3duIGNsYXNz
storagePrefix: myPrefix_
```

Exemple de configuration du backend avec nameTemplate

```
---
version: 1
storageDriverName: ontap-nas
backendName: ontap-nas-backend
managementLIF: <ip address>
svm: svm0
username: <admin>
password: <password>
defaults:
  nameTemplate:
"{{.volume.Name}}_{{.labels.cluster}}_{{.volume.Namespace}}_{{.vo\
    lume.RequestName}}"
labels:
  cluster: ClusterA
  PVC: "{{.volume.Namespace}}_{{.volume.RequestName}}"
```

Exemples de backends avec pools virtuels

Dans les exemples de fichiers de définition de backend ci-dessous, des valeurs par défaut spécifiques sont définies pour tous les pools de stockage, comme `spaceReserve` à aucun, `spaceAllocation` à faux, et `encryption` à faux. Les pools virtuels sont définis dans la section storage.

Trident définit les étiquettes de provisionnement dans le champ « Commentaires ». Les commentaires sont définis sur FlexVol pour `ontap-nas` ou sur FlexGroup pour `ontap-nas-flexgroup`. Trident copie toutes les

étiquettes présentes sur un pool virtuel vers le volume de stockage lors du provisionnement. Pour plus de simplicité, les administrateurs de stockage peuvent définir des étiquettes par pool virtuel et regrouper les volumes par étiquette.

Dans ces exemples, certains pools de stockage définissent leurs propres `spaceReserve`, `spaceAllocation`, et `encryption` valeurs, et certains pools remplacent les valeurs par défaut.

Exemple NAS ONTAP

```
---
version: 1
storageDriverName: ontap-nas
managementLIF: 10.0.0.1
svm: svm_nfs
username: admin
password: <password>
nfsMountOptions: nfsvers=4
defaults:
  spaceReserve: none
  encryption: "false"
  qosPolicy: standard
labels:
  store: nas_store
  k8scluster: prod-cluster-1
region: us_east_1
storage:
  - labels:
      app: msoffice
      cost: "100"
    zone: us_east_1a
    defaults:
      spaceReserve: volume
      encryption: "true"
      unixPermissions: "0755"
      adaptiveQosPolicy: adaptive-premium
  - labels:
      app: slack
      cost: "75"
    zone: us_east_1b
    defaults:
      spaceReserve: none
      encryption: "true"
      unixPermissions: "0755"
  - labels:
      department: legal
      creditpoints: "5000"
    zone: us_east_1b
    defaults:
      spaceReserve: none
      encryption: "true"
      unixPermissions: "0755"
  - labels:
      app: wordpress
```

```
      cost: "50"
    zone: us_east_1c
    defaults:
      spaceReserve: none
      encryption: "true"
      unixPermissions: "0775"
  - labels:
      app: mysqldb
      cost: "25"
    zone: us_east_1d
    defaults:
      spaceReserve: volume
      encryption: "false"
      unixPermissions: "0775"
```

Exemple de NAS ONTAP FlexGroup

```
---
version: 1
storageDriverName: ontap-nas-flexgroup
managementLIF: 10.0.0.1
svm: svm_nfs
username: vsadmin
password: <password>
defaults:
  spaceReserve: none
  encryption: "false"
labels:
  store: flexgroup_store
  k8scluster: prod-cluster-1
region: us_east_1
storage:
  - labels:
      protection: gold
      creditpoints: "50000"
    zone: us_east_1a
    defaults:
      spaceReserve: volume
      encryption: "true"
      unixPermissions: "0755"
  - labels:
      protection: gold
      creditpoints: "30000"
    zone: us_east_1b
    defaults:
      spaceReserve: none
      encryption: "true"
      unixPermissions: "0755"
  - labels:
      protection: silver
      creditpoints: "20000"
    zone: us_east_1c
    defaults:
      spaceReserve: none
      encryption: "true"
      unixPermissions: "0775"
  - labels:
      protection: bronze
      creditpoints: "10000"
    zone: us_east_1d
    defaults:
```

```
      spaceReserve: volume
      encryption: "false"
      unixPermissions: "0775"
```

Exemple d'économie NAS ONTAP

```
---
version: 1
storageDriverName: ontap-nas-economy
managementLIF: 10.0.0.1
svm: svm_nfs
username: vsadmin
password: <password>
defaults:
  spaceReserve: none
  encryption: "false"
labels:
  store: nas_economy_store
region: us_east_1
storage:
  - labels:
      department: finance
      creditpoints: "6000"
    zone: us_east_1a
    defaults:
      spaceReserve: volume
      encryption: "true"
      unixPermissions: "0755"
  - labels:
      protection: bronze
      creditpoints: "5000"
    zone: us_east_1b
    defaults:
      spaceReserve: none
      encryption: "true"
      unixPermissions: "0755"
  - labels:
      department: engineering
      creditpoints: "3000"
    zone: us_east_1c
    defaults:
      spaceReserve: none
      encryption: "true"
      unixPermissions: "0775"
  - labels:
      department: humanresource
      creditpoints: "2000"
    zone: us_east_1d
    defaults:
      spaceReserve: volume
```

```
      encryption: "false"
      unixPermissions: "0775"
```

Map backends vers StorageClasses

Les définitions de StorageClass suivantes font référence à Exemples de backends avec pools virtuels. En utilisant le champ `parameters.selector`, chaque StorageClass indique quels pools virtuels peuvent être utilisés pour héberger un volume. Le volume possédera les aspects définis dans le pool virtuel choisi.

- Le `protection-gold` StorageClass sera associé au premier et au deuxième pool virtuel dans le `ontap-nas-flexgroup` backend. Ce sont les seuls pools offrant une protection de niveau or.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-gold
provisioner: csi.trident.netapp.io
parameters:
  selector: "protection=gold"
  fsType: "ext4"
```

- Le `protection-not-gold` StorageClass sera associé au troisième et au quatrième pool virtuel dans le `ontap-nas-flexgroup` backend. Ce sont les seuls pools offrant un niveau de protection autre que gold.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-not-gold
provisioner: csi.trident.netapp.io
parameters:
  selector: "protection!=gold"
  fsType: "ext4"
```

- Le `app-mysqldb` StorageClass correspondra au quatrième pool virtuel dans le `ontap-nas` backend. Il s'agit du seul pool offrant une configuration de pool de stockage pour les applications de type mysqldb.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: app-mysqldb
provisioner: csi.trident.netapp.io
parameters:
  selector: "app=mysqldb"
  fsType: "ext4"
```

- Le `protection-silver-creditpoints-20k` StorageClass sera associé au troisième pool virtuel dans le `ontap-nas-flexgroup` backend. Il s'agit du seul pool offrant une protection de niveau argent et 20000 points de crédit.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-silver-creditpoints-20k
provisioner: csi.trident.netapp.io
parameters:
  selector: "protection=silver; creditpoints=20000"
  fsType: "ext4"
```

- Le `creditpoints-5k` StorageClass correspondra au troisième pool virtuel dans le `ontap-nas` backend et au deuxième pool virtuel dans le `ontap-nas-economy` backend. Ce sont les seuls pools proposant 5000 points de crédit.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: creditpoints-5k
provisioner: csi.trident.netapp.io
parameters:
  selector: "creditpoints=5000"
  fsType: "ext4"
```

Trident déterminera quel pool virtuel est sélectionné et s'assurera que l'exigence de stockage est respectée.

Mise à jour `dataLIF` après la configuration initiale

Vous pouvez modifier le dataLIF après la configuration initiale en exécutant la commande suivante pour fournir le nouveau fichier JSON backend avec le dataLIF mis à jour.

```
tridentctl update backend <backend-name> -f <path-to-backend-json-file-
with-updated-dataLIF>
```

REMARQUE	Si des PVC sont connectés à un ou plusieurs pods, vous devez mettre hors service tous les pods correspondants, puis les remettre en service afin que la nouvelle dataLIF prenne effet.

Exemples de SMB sécurisés

Configuration du backend avec le pilote ontap-nas

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-nas
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-nas
  managementLIF: 10.0.0.1
  svm: svm2
  nasType: smb
  defaults:
    adAdminUser: tridentADtest
  credentials:
    name: backend-tbc-ontap-invest-secret
```

Configuration du backend avec le pilote ontap-nas-economy

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-nas
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-nas-economy
  managementLIF: 10.0.0.1
  svm: svm2
  nasType: smb
  defaults:
    adAdminUser: tridentADtest
  credentials:
    name: backend-tbc-ontap-invest-secret
```

Configuration du backend avec pool de stockage

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-nas
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-nas
  managementLIF: 10.0.0.1
  svm: svm0
  useREST: false
  storage:
  - labels:
      app: msoffice
    defaults:
      adAdminUser: tridentADuser
  nasType: smb
  credentials:
    name: backend-tbc-ontap-invest-secret
```

Exemple de classe de stockage avec le pilote ontap-nas

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-smb-sc
  annotations:
    trident.netapp.io/smbShareAdUserPermission: change
    trident.netapp.io/smbShareAdUser: tridentADtest
parameters:
  backendType: ontap-nas
  csi.storage.k8s.io/node-stage-secret-name: smbcreds
  csi.storage.k8s.io/node-stage-secret-namespace: trident
  trident.netapp.io/nasType: smb
provisioner: csi.trident.netapp.io
reclaimPolicy: Delete
volumeBindingMode: Immediate
```

REMARQUE	Assurez-vous d'ajouter `annotations` pour activer le SMB sécurisé. Le SMB sécurisé ne fonctionne pas sans les annotations, quelles que soient les configurations définies dans le Backend ou le PVC.

Exemple de classe de stockage avec le pilote ontap-nas-economy

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-smb-sc
  annotations:
    trident.netapp.io/smbShareAdUserPermission: change
    trident.netapp.io/smbShareAdUser: tridentADuser3
parameters:
  backendType: ontap-nas-economy
  csi.storage.k8s.io/node-stage-secret-name: smbcreds
  csi.storage.k8s.io/node-stage-secret-namespace: trident
  trident.netapp.io/nasType: smb
provisioner: csi.trident.netapp.io
reclaimPolicy: Delete
volumeBindingMode: Immediate
```

Exemple de PVC avec un seul utilisateur AD

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-pvc4
  namespace: trident
  annotations:
    trident.netapp.io/smbShareAccessControl: |
      change:
        - tridentADtest
      read:
        - tridentADuser
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: ontap-smb-sc
```

Exemple de PVC avec plusieurs utilisateurs AD

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-test-pvc
  annotations:
    trident.netapp.io/smbShareAccessControl: |
      full_control:
        - tridentTestuser
        - tridentuser
        - tridentTestuser1
        - tridentuser1
      change:
        - tridentADuser
        - tridentADuser1
        - tridentADuser4
        - tridentTestuser2
      read:
        - tridentTestuser2
        - tridentTestuser3
        - tridentADuser2
        - tridentADuser3
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
```

Amazon FSx for NetApp ONTAP

Utilisez Trident avec Amazon FSx for NetApp ONTAP

"Amazon FSx for NetApp ONTAP" est un service AWS entièrement géré qui exécute des systèmes de fichiers alimentés par le système d'exploitation de stockage NetApp ONTAP. Il fournit les fonctionnalités, les performances et l'administration d'ONTAP avec l'évolutivité et la simplicité opérationnelle d'AWS. Un système de fichiers est la ressource principale dans Amazon FSx et est analogue à un cluster ONTAP sur site. Chaque système de fichiers contient une ou plusieurs machines virtuelles de stockage (SVM), et chaque SVM contient un ou plusieurs volumes qui stockent des fichiers et des répertoires. Cette intégration permet aux clusters Kubernetes exécutés dans Amazon Elastic Kubernetes Service (EKS) de provisionner des volumes persistants basés sur ONTAP pour les charges de travail bloc et fichier.

Exigences

En plus de "Exigences de Trident", pour intégrer FSx for ONTAP avec Trident, vous avez besoin de :

- Un cluster Amazon EKS existant ou un cluster Kubernetes autogéré avec `kubectl` installé.
- Un système de fichiers Amazon FSx for NetApp ONTAP existant et une machine virtuelle de stockage (SVM) accessibles depuis les nœuds de travail de votre cluster.
- Nœuds de travail préparés pour "NFS ou iSCSI".

REMARQUE	Assurez-vous de suivre les étapes de préparation des nœuds requises pour Amazon Linux et Ubuntu "Amazon Machine Images" (AMIs) en fonction de votre type d'AMI EKS.

Considérations

- Volumes SMB :
  - Les volumes SMB sont pris en charge à l'aide du pilote `ontap-nas` uniquement.
  - Les volumes SMB ne sont pas pris en charge avec l'add-on Trident EKS.
  - Trident prend uniquement en charge les volumes SMB montés sur des pods exécutés sur des nœuds Windows. Consultez "Préparez-vous à provisionner des volumes SMB" pour plus de détails.
- Avant Trident 24.02, les volumes créés sur les systèmes de fichiers Amazon FSx avec les sauvegardes automatiques activées ne pouvaient pas être supprimés par Trident. Pour éviter ce problème dans Trident 24.02 ou version ultérieure, spécifiez le `fsxFilesystemID`, AWS `apiRegion`, AWS `apikey`, et AWS `secretKey` dans le fichier de configuration du backend pour AWS FSx for ONTAP.

REMARQUE	Si vous spécifiez un rôle IAM à Trident, vous pouvez omettre de spécifier les champs `apiRegion`, `apiKey`, et `secretKey` à Trident explicitement. Pour plus d'informations, consultez "Options de configuration et exemples pour FSx for ONTAP".

Utilisation simultanée de Trident SAN/iSCSI et du pilote EBS-CSI

Si vous prévoyez d'utiliser des pilotes ontap-san (par exemple, iSCSI) avec AWS (EKS, ROSA, EC2 ou toute autre instance), la configuration multipath requise sur les nœuds peut entrer en conflit avec le pilote CSI Amazon Elastic Block Store (EBS). Pour garantir que le multipath fonctionne sans interférer avec les disques EBS sur le même nœud, vous devez exclure EBS de votre configuration multipath. Cet exemple montre un fichier `multipath.conf` qui inclut les paramètres Trident requis tout en excluant les disques EBS du multipath :

```
defaults {
    find_multipaths no
}
blacklist {
    device {
        vendor "NVME"
        product "Amazon Elastic Block Store"
    }
}
```

Authentification

Trident propose deux modes d'authentification.

- Authentification par identifiants (recommandée) : stocke les identifiants en toute sécurité dans AWS Secrets Manager. Vous pouvez utiliser l `fsxadmin` utilisateur de votre système de fichiers ou l `vsadmin` utilisateur configuré pour votre SVM.

AVERTISSEMENT	Trident s'attend à être exécuté en tant qu' `vsadmin`utilisateur SVM ou en tant qu'utilisateur portant un nom différent mais ayant le même rôle. Amazon FSx for NetApp ONTAP dispose d'un `fsxadmin` utilisateur qui est un remplacement limité de l'utilisateur du cluster ONTAP `admin`. Nous recommandons fortement d'utiliser `vsadmin` avec Trident.

- Basé sur un certificat : Trident communiquera avec la SVM de votre système de fichiers FSx à l'aide d'un certificat installé sur votre SVM.

Pour plus d'informations sur l'activation de l'authentification, reportez-vous à l'authentification pour votre type de pilote :

- "Authentification NAS ONTAP"
- "Authentification SAN ONTAP"

Amazon Machine Images (AMIs) testées

Le cluster EKS prend en charge différents systèmes d'exploitation, mais AWS a optimisé certaines images machine Amazon (AMIs) pour les conteneurs et EKS. Les AMIs suivantes ont été testées avec NetApp Trident 25.02.

AMI	NAS	NAS-economy	iSCSI	iSCSI-economy
AL2023_x86_64_STANDARD	Oui	Oui	Oui	Oui
AL2_x86_64	Oui	Oui	Oui*	Oui*
BOTTLEROCKET_x86_64	Oui**	Oui	S/O	S/O

AL2023_ARM_64_STANDARD	Oui	Oui	Oui	Oui
AL2_ARM_64	Oui	Oui	Oui*	Oui*
BOTTLEROCKET_ARM_64	Oui**	Oui	S/O	S/O

- * Impossible de supprimer le PV sans redémarrer le nœud
- ** Ne fonctionne pas avec NFSv3 avec Trident version 25.02.

REMARQUE	Si l'AMI que vous recherchez ne figure pas dans cette liste, cela ne signifie pas qu'elle n'est pas prise en charge ; cela signifie simplement qu'elle n'a pas été testée. Cette liste sert de guide pour les AMI dont le fonctionnement est avéré.

Tests effectués avec:

- EKS version : 1.32
- Méthode d'installation : Helm 25.06 et en tant qu'AWS add-On 25.06
- Pour NAS, les protocoles NFSv3 et NFSv4.1 ont été testés.
- Pour le SAN, seul l'iSCSI a été testé, pas le NVMe-oF.

Tests effectués :

- Créer : Storage Class, pvc, pod
- Supprimer : pod, pvc (standard, qtree/lun – économique, NAS avec sauvegarde AWS)

Pour plus d'informations

- "Documentation Amazon FSx for NetApp ONTAP"
- "Article de blog sur Amazon FSx for NetApp ONTAP"

Créez un rôle IAM et un secret AWS

Vous pouvez configurer les pods Kubernetes pour accéder aux ressources AWS en s'authentifiant en tant que rôle AWS IAM au lieu de fournir des identifiants AWS explicites.

REMARQUE	Pour vous authentifier à l'aide d'un rôle AWS IAM, vous devez disposer d'un cluster Kubernetes déployé à l'aide d'EKS.

Créer un secret AWS Secrets Manager

Étant donné que Trident utilisera des API sur un vserver FSx pour gérer le stockage pour vous, il aura besoin d'identifiants pour le faire. La méthode sécurisée pour transmettre ces identifiants consiste à utiliser un secret AWS Secrets Manager. Par conséquent, si vous n'en possédez pas déjà un, vous devrez créer un secret AWS Secrets Manager contenant les identifiants du compte vsadmin.

Cet exemple crée un secret AWS Secrets Manager pour stocker les informations d'identification Trident CSI :

```
aws secretsmanager create-secret --name trident-secret --description
"Trident CSI credentials"\
  --secret-string
"{\"username\":\"vsadmin\",\"password\":\"<svmpassword>\"}"
```

Créer une stratégie IAM

Trident a également besoin des autorisations AWS pour fonctionner correctement. Par conséquent, vous devez créer une stratégie qui donne à Trident les autorisations dont il a besoin.

Les exemples suivants créent une stratégie IAM à l'aide de l'AWS CLI :

```
aws iam create-policy --policy-name AmazonFSxNCSIDriverPolicy --policy
-document file://policy.json
  --description "This policy grants access to Trident CSI to FSxN and
Secrets manager"
```

Exemple de JSON de stratégie :

```
{
  "Statement": [
    {
      "Action": [
        "fsx:DescribeFileSystems",
        "fsx:DescribeVolumes",
        "fsx:CreateVolume",
        "fsx:RestoreVolumeFromSnapshot",
        "fsx:DescribeStorageVirtualMachines",
        "fsx:UntagResource",
        "fsx:UpdateVolume",
        "fsx:TagResource",
        "fsx:DeleteVolume"
      ],
      "Effect": "Allow",
      "Resource": "*"
    },
    {
      "Action": "secretsmanager:GetSecretValue",
      "Effect": "Allow",
      "Resource": "arn:aws:secretsmanager:<aws-region>:<aws-account-
id>:secret:<aws-secret-manager-name>*"
    }
  ],
  "Version": "2012-10-17"
}
```

Créer une identité de pod ou un rôle IAM pour l'association du compte de service (IRSA)

Vous pouvez configurer un compte de service Kubernetes pour qu'il assume un rôle AWS Identity and Access Management (IAM) avec EKS Pod Identity ou IAM role for Service account association (IRSA). Tous les Pods configurés pour utiliser le compte de service peuvent alors accéder à tout service AWS auquel le rôle a des autorisations d'accès.

Identité du pod

Les associations d'identité de pod Amazon EKS offrent la possibilité de gérer les informations d'identification de vos applications, de la même manière que les profils d'instance Amazon EC2 fournissent des informations d'identification aux instances Amazon EC2.

Installez Pod Identity sur votre cluster EKS :

Vous pouvez créer une identité de pod via la console AWS ou en utilisant la commande AWS CLI suivante :

```
aws eks create-addon --cluster-name <EKS_CLUSTER_NAME> --addon-name
eks-pod-identity-agent
```

Pour plus d'informations, consultez "Configurer l'agent d'identité du pod Amazon EKS".

Créer trust-relationship.json:

Créez trust-relationship.json pour permettre au Service Principal EKS d'assumer ce rôle pour l'identité du pod. Créez ensuite un rôle avec cette stratégie d'approbation :

```
aws iam create-role \
  --role-name fsxn-csi-role --assume-role-policy-document file://trust-
relationship.json \
  --description "fsxn csi pod identity role"
```

Fichier trust-relationship.json:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "pods.eks.amazonaws.com"
      },
      "Action": [
        "sts:AssumeRole",
        "sts:TagSession"
      ]
    }
  ]
}
```

Associez la stratégie de rôle au rôle IAM:

Associez la stratégie de rôle de l'étape précédente au rôle IAM qui a été créé :

```
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:111122223333:policy/fsxn-csi-policy \
  --role-name fsxn-csi-role
```

Créer une association d'identité de pod :

Créer une association d'identité de pod entre le rôle IAM et le compte de service Trident (trident-controller)

```
aws eks create-pod-identity-association \
  --cluster-name <EKS_CLUSTER_NAME> \
  --role-arn arn:aws:iam::111122223333:role/fsxn-csi-role \
  --namespace trident --service-account trident-controller
```

Rôle IAM pour l'association de compte de service (IRSA)

Utilisation de l'interface de ligne de commande AWS :

```
aws iam create-role --role-name AmazonEKS_FSxN_CSI_DriverRole \
  --assume-role-policy-document file://trust-relationship.json
```

Fichier trust-relationship.json :

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws:iam::<account_id>:oidc-
provider/<oidc_provider>"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringEquals": {
          "<oidc_provider>:aud": "sts.amazonaws.com",
          "<oidc_provider>:sub":
"system:serviceaccount:trident:trident-controller"
        }
      }
    }
  ]
}
```

Mettez à jour les valeurs suivantes dans le fichier `trust-relationship.json` :

- **<account_id>** - Votre ID de compte AWS
- **<oidc_provider>** - L'OIDC de votre cluster EKS. Vous pouvez obtenir le oidc_provider en exécutant :

```
aws eks describe-cluster --name my-cluster --query
"cluster.identity.oidc.issuer"\
  --output text | sed -e "s/^https:\/\///"
```

Associez le rôle IAM à la stratégie IAM:

Une fois le rôle créé, associez la stratégie (qui a été créée à l'étape ci-dessus) au rôle à l'aide de cette commande :

```
aws iam attach-role-policy --role-name my-role --policy-arn <IAM policy
ARN>
```

Vérifiez que le fournisseur OICD est associé:

Vérifiez que votre fournisseur OIDC est associé à votre cluster. Vous pouvez le vérifier à l'aide de cette commande :

```
aws iam list-open-id-connect-providers | grep $oidc_id | cut -d "/" -f4
```

Si le résultat est vide, utilisez la commande suivante pour associer IAM OIDC à votre cluster :

```
eksctl utils associate-iam-oidc-provider --cluster $cluster_name
--approve
```

Si vous utilisez eksctl, utilisez l'exemple suivant pour créer un rôle IAM pour le compte de service dans EKS :

```
eksctl create iamserviceaccount --name trident-controller --namespace
trident \
  --cluster <my-cluster> --role-name AmazonEKS_FSxN_CSI_DriverRole
--role-only \
  --attach-policy-arn <IAM-Policy ARN> --approve
```

Installer Trident

Trident simplifie la gestion du stockage Amazon FSx for NetApp ONTAP dans Kubernetes afin de permettre à vos développeurs et administrateurs de se concentrer sur le déploiement des applications. Vous pouvez installer Trident en utilisant l'une des méthodes suivantes :

- Helm
- Module complémentaire EKS

Si vous souhaitez utiliser la fonctionnalité de snapshot, installez l'add-on CSI snapshot controller. Consultez "Activer la fonctionnalité de snapshot pour les volumes CSI" pour plus d'informations.

Installez Trident via helm

Identité du pod

1. Ajoutez le dépôt Trident Helm :

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

2. Installez Trident en utilisant l'exemple suivant :

```
helm install trident-operator netapp-trident/trident-operator --version 100.2502.1 --namespace trident --create-namespace
```

Vous pouvez utiliser la commande `helm list` pour consulter les détails de l'installation tels que le nom, l'espace de noms, le chart, l'état, la version de l'application et le numéro de révision.

```
helm list -n trident
```

```
NAME                NAMESPACE   REVISION    UPDATED
STATUS       CHART                          APP VERSION

trident-operator    trident     1           2024-10-14
14:31:22.463122 +0300 IDT    deployed     trident-operator-
100.2502.0    25.02.0
```

Association de compte de service (IRSA)

1. Ajoutez le dépôt Trident Helm :

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

2. Définissez les valeurs pour **fournisseur de cloud** et **cloud identity** :

```
helm install trident-operator netapp-trident/trident-operator
--version 100.2502.1  \
--set cloudProvider="AWS" \
--set cloudIdentity="'eks.amazonaws.com/role-arn:
arn:aws:iam::<accountID>:role/<AmazonEKS_FSxN_CSI_DriverRole>'" \
--namespace trident \
--create-namespace
```

Vous pouvez utiliser la commande `helm list` pour consulter les détails de l'installation tels que le nom, l'espace de noms, le chart, l'état, la version de l'application et le numéro de révision.

```
helm list -n trident
```

```
NAME                NAMESPACE   REVISION    UPDATED
STATUS       CHART                          APP VERSION

trident-operator    trident     1           2024-10-14
14:31:22.463122 +0300 IDT    deployed     trident-operator-
100.2510.0    25.10.0
```

REMARQUE

Si vous prévoyez d'utiliser iSCSI, assurez-vous que iSCSI est activé sur votre machine cliente. Si vous utilisez AL2023 Worker node OS, vous pouvez automatiser l'installation du client iSCSI en ajoutant le paramètre node prep lors de l'installation avec helm :

```
helm install trident-operator netapp-trident/trident-
operator --version 100.2502.1 --namespace trident --create
-namespace --set nodePrep={iscsi}
```

Installez Trident via le module complémentaire EKS

L'add-on Trident EKS inclut les derniers correctifs de sécurité, les corrections de bogues et est validé par AWS pour fonctionner avec Amazon EKS. L'add-on EKS vous permet de garantir de manière cohérente que vos clusters Amazon EKS sont sécurisés et stables, et de réduire la quantité de travail nécessaire pour installer, configurer et mettre à jour les add-ons.

Prérequis

Assurez-vous de disposer des éléments suivants avant de configurer le module complémentaire Trident pour AWS EKS :

- Un compte de cluster Amazon EKS avec abonnement complémentaire
- Autorisations AWS pour la AWS marketplace :
  `"aws-marketplace:ViewSubscriptions",`
  `"aws-marketplace:Subscribe",`
  `"aws-marketplace:Unsubscribe`
- Type d'AMI : Amazon Linux 2 (AL2_x86_64) ou Amazon Linux 2 Arm(AL2_ARM_64)
- Type de nœud : AMD ou ARM
- Un système de fichiers Amazon FSx for NetApp ONTAP existant

Activez le module complémentaire Trident pour AWS

Console de gestion

1. Ouvrez la console Amazon EKS à https://console.aws.amazon.com/eks/home#/clusters.
2. Dans le volet de navigation de gauche, sélectionnez **Clusters**.
3. Sélectionnez le nom du cluster pour lequel vous souhaitez configurer le module complémentaire NetApp Trident CSI.
4. Sélectionnez **Modules complémentaires** puis sélectionnez **Obtenir plus de modules complémentaires**.
5. Suivez ces étapes pour sélectionner le module complémentaire :
   a. Faites défiler vers le bas jusqu'à la section **AWS Marketplace add-ons** et tapez **"Trident"** dans la zone de recherche.
   b. Cochez la case à cocher située dans le coin supérieur droit de la boîte Trident by NetApp.
   c. Sélectionnez **Next**.
6. Sur la page des paramètres **Configure selected add-ons**, procédez comme suit :

REMARQUE	Ignorez ces étapes si vous utilisez l'association Pod Identity.

   a. Sélectionnez la **Version** que vous souhaitez utiliser.
   b. Si vous utilisez l'authentification IRSA, assurez-vous de définir les valeurs de configuration disponibles dans les paramètres de configuration optionnels :
      - Sélectionnez la **Version** que vous souhaitez utiliser.
      - Suivez le **schéma de configuration du module complémentaire** et définissez le paramètre **configurationValues** dans la section **Valeurs de configuration** sur le role-arn que vous avez créé à l'étape précédente (la valeur doit être au format suivant) :

```
{

  "cloudIdentity": "'eks.amazonaws.com/role-arn: <role ARN>'",
  "cloudProvider": "AWS"

}
```

   +
   Si vous sélectionnez Override comme méthode de résolution des conflits, un ou plusieurs des paramètres de l'add-on existant peuvent être remplacés par les paramètres de l'add-on Amazon EKS. Si vous n'activez pas cette option et qu'il y a un conflit avec vos paramètres existants, l'opération échoue. Vous pouvez utiliser le message d'erreur résultant pour résoudre le conflit. Avant de sélectionner cette option, assurez-vous que l'add-on Amazon EKS ne gère pas des paramètres que vous devez gérer vous-même.

7. Choisissez **Suivant**.
8. Sur la page **Vérifier et ajouter**, choisissez **Créer**.

   Une fois l'installation du logiciel complémentaire terminée, vous voyez votre logiciel complémentaire installé.

AWS CLI

1. Créez le `add-on.json` fichier :

Pour l'identité du pod, utilisez le format suivant:

REMARQUE Utilisez le

```
{
  "clusterName": "<eks-cluster>",
  "addonName": "netapp_trident-operator",
  "addonVersion": "v25.6.0-eksbuild.1",
}
```

Pour l'authentification IRSA, utilisez le format suivant:

```
{
  "clusterName": "<eks-cluster>",
  "addonName": "netapp_trident-operator",
  "addonVersion": "v25.6.0-eksbuild.1",
  "serviceAccountRoleArn": "<role ARN>",
  "configurationValues": {
    "cloudIdentity": "'eks.amazonaws.com/role-arn: <role ARN>'",
    "cloudProvider": "AWS"
  }
}
```

REMARQUE Remplacez `<role ARN>` par l'ARN du rôle qui a été créé à l'étape précédente.

2. Installez le logiciel complémentaire Trident EKS.

```
aws eks create-addon --cli-input-json file://add-on.json
```

eksctl

La commande suivante permet d'installer le module complémentaire Trident EKS :

```
eksctl create addon --name netapp_trident-operator --cluster
<cluster_name> --force
```

Mettre à jour le module complémentaire Trident EKS

Console de gestion

1. Ouvrez la console Amazon EKS https://console.aws.amazon.com/eks/home#/clusters.
2. Dans le volet de navigation de gauche, sélectionnez **Clusters**.
3. Sélectionnez le nom du cluster pour lequel vous souhaitez mettre à jour le logiciel complémentaire NetApp Trident CSI.
4. Sélectionnez l'onglet **Add-ons**.
5. Sélectionnez **Trident par NetApp** puis sélectionnez **Modifier**.
6. Sur la page **Configurer Trident par NetApp**, procédez comme suit :
   a. Sélectionnez la **Version** que vous souhaitez utiliser.
   b. Développez les **Paramètres de configuration optionnels** et modifiez-les si nécessaire.
   c. Sélectionnez **Save changes**.

AWS CLI

L'exemple suivant met à jour le add-on EKS :

```
aws eks update-addon --cluster-name <eks_cluster_name> --addon-name
netapp_trident-operator --addon-version v25.6.0-eksbuild.1 \
  --service-account-role-arn <role-ARN> --resolve-conflict preserve \
  --configuration-values "{\"cloudIdentity\":
\"'eks.amazonaws.com/role-arn: <role ARN>'\"}"
```

eksctl

- Vérifiez la version actuelle de votre add-on Trident CSI FSxN. Remplacez `my-cluster` par le nom de votre cluster.

  ```
  eksctl get addon --name netapp_trident-operator --cluster my-cluster
  ```

  Exemple de sortie :

```
NAME                          VERSION              STATUS    ISSUES
IAMROLE    UPDATE AVAILABLE    CONFIGURATION VALUES
netapp_trident-operator    v25.6.0-eksbuild.1    ACTIVE    0
{"cloudIdentity":"'eks.amazonaws.com/role-arn:
arn:aws:iam::139763910815:role/AmazonEKS_FSXN_CSI_DriverRole'"}
```

- Mettez à jour le logiciel complémentaire avec la version renvoyée sous UPDATE AVAILABLE dans le résultat de l'étape précédente.

  ```
  eksctl update addon --name netapp_trident-operator --version
  v25.6.0-eksbuild.1 --cluster my-cluster --force
  ```

Si vous supprimez l' `--force`option et que l'un des paramètres du module complémentaire Amazon EKS entre en conflit avec vos paramètres existants, la mise à jour du module complémentaire Amazon EKS échoue ; vous recevez un message d'erreur pour vous aider à résoudre le conflit. Avant de spécifier cette option, assurez-vous que le module complémentaire Amazon EKS ne gère pas de paramètres que vous devez gérer, car ces paramètres sont écrasés avec cette option. Pour plus d'informations sur les autres options de ce paramètre, consultez "Modules complémentaires". Pour plus d'informations sur la gestion des champs Kubernetes d'Amazon EKS, consultez "Gestion des champs Kubernetes".

Désinstallez/supprimez le module complémentaire Trident EKS

Vous avez deux options pour supprimer un add-on Amazon EKS :

- **Conserver le logiciel complémentaire sur votre cluster** – Cette option supprime la gestion de tous les paramètres par Amazon EKS. Elle supprime également la capacité d'Amazon EKS à vous notifier des mises à jour et à mettre à jour automatiquement le logiciel complémentaire Amazon EKS après que vous ayez initié une mise à jour. Cependant, elle conserve le logiciel complémentaire sur votre cluster. Cette option fait du logiciel complémentaire une installation autogérée, plutôt qu'un logiciel complémentaire Amazon EKS. Avec cette option, il n'y a pas d'interruption pour le logiciel complémentaire. Conservez l' `--preserve`option dans la commande pour préserver le logiciel complémentaire.
- **Supprimez entièrement le logiciel complémentaire de votre cluster** – NetApp recommande de supprimer le module complémentaire Amazon EKS de votre cluster uniquement s'il n'y a aucune ressource sur votre cluster qui en dépend. Supprimez l'option `--preserve` du `delete`commande pour supprimer le logiciel complémentaire.

REMARQUE	Si le logiciel complémentaire a un compte IAM associé, le compte IAM n'est pas supprimé.

Console de gestion

1. Ouvrez la console Amazon EKS à https://console.aws.amazon.com/eks/home#/clusters.
2. Dans le volet de navigation de gauche, sélectionnez **Clusters**.
3. Sélectionnez le nom du cluster pour lequel vous souhaitez supprimer le logiciel complémentaire Trident CSI NetApp.
4. Sélectionnez l'onglet **Add-ons** puis sélectionnez **Trident par NetApp**.*
5. Sélectionnez **Remove**.
6. Dans la boîte de dialogue **Remove netapp_trident-operator confirmation**, procédez comme suit :
   a. Si vous souhaitez qu'Amazon EKS cesse de gérer les paramètres du logiciel complémentaire, sélectionnez **Conserver sur le cluster**. Procédez ainsi si vous souhaitez conserver le logiciel complémentaire sur votre cluster afin de pouvoir gérer vous-même tous les paramètres du logiciel complémentaire.
   b. Saisissez **netapp_trident-operator**.
   c. Sélectionnez **Remove**.

AWS CLI

Remplacez `my-cluster` par le nom de votre cluster, puis exécutez la commande suivante.

```
aws eks delete-addon --cluster-name my-cluster --addon-name
netapp_trident-operator --preserve
```

eksctl

La commande suivante désinstalle le module complémentaire Trident EKS :

```
eksctl delete addon --cluster K8s-arm --name netapp_trident-operator
```

Configurer une classe de stockage

La "Objet Kubernetes StorageClass" identifie un provisionneur et lui indique comment provisionner les volumes. Cette section montre comment configurer un objet Kubernetes StorageClass qui spécifie Trident comme provisionneur.

Créer un objet StorageClass

Lorsque vous créez un StorageClass pour FSx for ONTAP, Trident créera automatiquement la configuration backend.

REMARQUE	Si vous souhaitez configurer manuellement le stockage backend, veuillez vous référer à la section [create-a-kubernetes-storageclass-without-automatic-backend-configuration] pour savoir comment créer séparément le backend Trident et la classe de stockage.

Spécifiez les paramètres requis StorageClass

Les trois paramètres suivants doivent être définis lors de la création d'un StorageClass :

Paramètre	Obligatoire	Type	Description
`fsxFilesystemID`	Oui	chaîne	ID du système de fichiers FSx for NetApp ONTAP
`storageDriverName`	Oui	chaîne	Pilote de stockage Trident (par exemple, `ontap-nas` ou `ontap-san`)
`credentialsName`	Oui	chaîne	Nom du secret Kubernetes qui contient les informations d'identification FSx for ONTAP

Spécifiez les paramètres optionnels

Vous pouvez transmettre des paramètres backend optionnels via le StorageClass. Définissez toutes les valeurs optionnelles sous forme de chaînes dans la section StorageClass `parameters`. Pour une liste complète des paramètres du backend, voir : "Configuration du backend FSx for NetApp ONTAP".

Exemples de fichiers de configuration StorageClass.

L'exemple suivant montre un StorageClass qui déclenche une configuration automatique du backend.

YAML

```yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-fsx-demo
  annotations:
    description: "Demo StorageClass for FSx for NetApp ONTAP"
provisioner: csi.trident.netapp.io
parameters:
  fsxFilesystemID: "fs-0abc123"
  storageDriverName: "ontap-nas"
  credentialsName: trident-fsx-credentials
allowVolumeExpansion: true
reclaimPolicy: Delete
volumeBindingMode: Immediate
```

JSON

```json
{
  "apiVersion": "storage.k8s.io/v1",
  "kind": "StorageClass",
  "metadata": {
    "name": "ontap-fsx-demo",
    "annotations": {
      "description": "Demo StorageClass for FSx for NetApp ONTAP"
    }
  },
  "provisioner": "csi.trident.netapp.io",
  "parameters": {
    "fsxFilesystemID": "fs-0abc123",
    "storageDriverName": "ontap-nas",
    "credentialsName": "trident-fsx-credentials"
  },
  "allowVolumeExpansion": true,
  "reclaimPolicy": "Delete",
  "volumeBindingMode": "Immediate"
}
```

Créez le StorageClass

Une fois que vous avez créé votre fichier de configuration, exécutez la commande suivante pour créer la classe de stockage.

```
kubectl create -f storage-class-ontapnas.yaml
```

Vous devriez maintenant voir une classe de stockage **basic-csi** dans Kubernetes et Trident, et Trident devrait avoir découvert les pools sur le backend.

```
kubectl get sc basic-csi
```

```
NAME         PROVISIONER             AGE
basic-csi    csi.trident.netapp.io   15h
```

Après avoir appliqué le StorageClass, Trident crée automatiquement le backend. Vous pouvez ensuite créer des PersistentVolumeClaims qui référencent ce StorageClass.

Vérifier l'état de la configuration du backend

Trident enregistre le résultat de la création du backend dans les annotations StorageClass.

Annotation	Description
`trident.netapp.io/configuratorStatus`	Résultat de la configuration (`Success` ou `Failure`)
`trident.netapp.io/configuratorMessage`	Message d'état ou d'erreur détaillé
`trident.netapp.io/configuratorName`	Nom de la ressource du configurateur interne
`trident.netapp.io/managed`	Indique que le StorageClass est géré par Trident
`trident.netapp.io/additionalStoragePools`	Pools de stockage créés pour ce backend

Pour vérifier l'état, exécutez :

```
kubectl get storageclass ontap-fsx-demo -o yaml
```

Confirmez que `trident.netapp.io/configuratorStatus` est défini sur `Success`. Si la valeur est `Failure`, examinez `trident.netapp.io/configuratorMessage` pour l'erreur.

Ajouter des systèmes de fichiers FSxN supplémentaires

Si vous avez besoin de stockage supplémentaire tout en continuant à utiliser le même StorageClass, ajoutez des identifiants de système de fichiers FSxN supplémentaires.

Modifiez le StorageClass et ajoutez l'annotation suivante :

```
metadata:
  annotations:
    trident.netapp.io/additionalFsxNFileSystemID: '["fs-
xxxxxxxxxxxxxxxxxx"]'
```

Une fois la modification appliquée, Trident met à jour la configuration du backend et met à jour les annotations StorageClass.

Considérations et limitations opérationnelles

- La suppression d'un StorageClass qui utilise la configuration automatique du backend supprime généralement le backend Trident associé. Cela peut perturber la connectivité du stockage et interrompre les charges de travail en cours. Validez l'impact avant de supprimer un StorageClass géré.
- La configuration automatique du backend est prise en charge uniquement pour AWS FSx pour NetApp ONTAP.

Créer un StorageClass Kubernetes sans configuration automatique du backend

Si vous souhaitez créer le backend Trident et le StorageClass séparément, suivez ces étapes.

Comprendre le fonctionnement de la configuration automatique du backend

Trident déduit la configuration du backend à partir de la définition de la StorageClass. Lorsque vous appliquez la StorageClass, Trident valide les paramètres requis, crée le backend et annote la StorageClass avec le statut.

Trident crée le VolumeSnapshotClass une seule fois. Trident réutilise le même VolumeSnapshotClass pour les StorageClasses suivants.

Créer le backend Trident

Pour créer un backend Trident, vous devez créer un fichier de configuration au format JSON ou YAML. Le fichier doit spécifier le type de stockage souhaité (NAS ou SAN), le système de fichiers, la SVM à utiliser et la méthode d'authentification. L'exemple suivant montre comment définir un stockage basé sur NAS et utiliser un secret AWS pour stocker les informations d'identification de la SVM que vous souhaitez utiliser :

YAML

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-nas
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-nas
  backendName: tbc-ontap-nas
  svm: svm-name
  aws:
    fsxFilesystemID: fs-xxxxxxxxxx
  credentials:
    name: "arn:aws:secretsmanager:us-west-2:xxxxxxxx:secret:secret-
name"
    type: awsarn
```

JSON

```
{
  "apiVersion": "trident.netapp.io/v1",
  "kind": "TridentBackendConfig",
  "metadata": {
    "name": "backend-tbc-ontap-nas"
    "namespace": "trident"
  },
  "spec": {
    "version": 1,
    "storageDriverName": "ontap-nas",
    "backendName": "tbc-ontap-nas",
    "svm": "svm-name",
    "aws": {
      "fsxFilesystemID": "fs-xxxxxxxxxx"
    },
    "managementLIF": null,
    "credentials": {
      "name": "arn:aws:secretsmanager:us-west-2:xxxxxxxx:secret:secret-
name",
      "type": "awsarn"
    }
  }
}
```

Détails du pilote FSx for ONTAP

Vous pouvez intégrer Trident à Amazon FSx for NetApp ONTAP à l'aide des pilotes suivants :

Nom du pilote	Description
`ontap-san`	Chaque PV provisionné est un LUN au sein de son propre volume Amazon FSx for NetApp ONTAP. Recommandé pour le stockage bloc.
`ontap-nas`	Chaque PV provisionné est un volume Amazon FSx for NetApp ONTAP complet. Recommandé pour NFS et SMB.
`ontap-san-economy`	Chaque PV provisionné est un LUN avec un nombre configurable de LUN par Amazon FSx for NetApp ONTAP volume.
`ontap-nas-economy`	Chaque PV provisionné est un qtree, avec un nombre configurable de qtrees par volume Amazon FSx for NetApp ONTAP.
`ontap-nas-flexgroup`	Chaque PV provisionné est un volume Amazon FSx for NetApp ONTAP FlexGroup complet.

Pour plus de détails sur le conducteur, consultez "Pilotes NAS" et "Pilotes SAN".

Créer le backend

Après avoir créé le fichier de configuration, exécutez les commandes suivantes pour créer et valider la configuration du backend Trident (TBC) :

- Créez une configuration backend Trident (TBC) à partir d'un fichier yaml et exécutez la commande suivante :

```
kubectl create -f backendconfig.yaml -n trident
```

```
tridentbackendconfig.trident.netapp.io/backend-tbc-ontap-nas created
```

- Vérifiez que la configuration du backend trident (TBC) a été créée avec succès :

```
Kubectl get tbc -n trident
```

```
NAME                          BACKEND NAME          BACKEND UUID
PHASE   STATUS

backend-tbc-ontap-nas         tbc-ontap-nas         933e0071-66ce-4324-
b9ff-f96d916ac5e9   Bound   Success
```

Pour plus d'informations sur les autres options de configuration, consultez la [Backend-advanced-configuration-and-examples] section ci-dessous.

Configurer une classe de stockage sans configuration automatique du backend

Voici des exemples de configurations de classes de stockage à utiliser avec Trident et FSx pour ONTAP.

Classe de stockage pour NFS

Vous pouvez utiliser cet exemple pour configurer StorageClass pour des volumes utilisant NFS (reportez-vous à la section Attributs Trident ci-dessous pour la liste complète des attributs) :

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-gold
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-nas"
  provisioningType: "thin"
  snapshots: "true"
```

Classe de stockage pour iSCSI

Utilisez cet exemple pour configurer StorageClass pour des volumes utilisant iSCSI :

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-gold
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
  provisioningType: "thin"
  snapshots: "true"
```

Classe de stockage utilisant NFSv3 et AWS Bottlerocket

Pour provisionner des volumes NFSv3 sur AWS Bottlerocket, ajoutez les `mountOptions` requis à la classe de stockage :

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-gold
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-nas"
  media: "ssd"
  provisioningType: "thin"
  snapshots: "true"
mountOptions:
  - nfsvers=3
  - nolock
```

Attributs de Trident StorageClass

Ces paramètres déterminent quels pools de stockage gérés par Trident doivent être utilisés pour provisionner des volumes d'un type donné.

Attribut	Type	Valeurs	Offre	Demande	Soutenu par
médias[1]	chaîne	hdd, hybride, ssd	Le pool contient des médias de ce type ; hybride signifie les deux	Type de média spécifié	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, solidfire-san
provisioningType	chaîne	mince, épais	Pool prend en charge cette méthode de provisionnement	Méthode de provisionnement spécifiée	épais : all ontap ; mince : all ontap & solidfire-san
backendType	chaîne	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, solidfire-san, azure-netapp-files, ontap-san-economy	Pool appartient à ce type de backend	Backend spécifié	Tous les drivers
instantanés	bool	vrai, faux	Pool prend en charge les volumes avec snapshots	Volume avec instantanés activés	ontap-nas, ontap-san, solidfire-san

Attribut	Type	Valeurs	Offre	Demande	Soutenu par
clones	bool	vrai, faux	Pool prend en charge le clonage des volumes	Volume avec clones activés	ontap-nas, ontap-san, solidfire-san
chiffrement	bool	vrai, faux	Pool prend en charge les volumes chiffrés	Volume avec chiffrement activé	ontap-nas, ontap-nas-economy, ontap-nas-flexgroups, ontap-san
Op E/S par sec	int	entier positif	Pool est capable de garantir des IOPS dans cette plage	Volume garanti ces IOPS	solidfire-san

[1] : Non pris en charge par ONTAP Select ou les systèmes FSx for ONTAP

Consultez "Objets Kubernetes et Trident" pour plus de détails sur la manière dont les classes de stockage interagissent avec le `PersistentVolumeClaim` et les paramètres permettant de contrôler la façon dont Trident provisionne les volumes.

Créer la classe de stockage

Une fois que vous avez configuré le StorageClass, vous pouvez le créer dans Kubernetes.

Étapes

1. Il s'agit d'un objet Kubernetes, utilisez donc `kubectl` pour le créer dans Kubernetes.

```
kubectl create -f storage-class-ontapnas.yaml
```

2. Vous devriez maintenant voir une classe de stockage **basic-csi** dans Kubernetes et Trident, et Trident devrait avoir découvert les pools sur le backend.

```
kubectl get sc basic-csi
```

```
NAME         PROVISIONER             AGE
basic-csi    csi.trident.netapp.io   15h
```

Provisionner des volumes SMB

Vous pouvez provisionner des volumes SMB à l'aide du `ontap-nas` driver. Cependant, pour ce faire, vous devez effectuer ces étapes : "Préparez-vous à provisionner des volumes SMB".

Configuration avancée du backend et exemples

Consultez le tableau suivant pour les options de configuration du backend :

Paramètre	Description	Exemple
`version`		Toujours 1
`storageDriverName`	Nom du pilote de stockage	`ontap-nas`, `ontap-nas-economy`, `ontap-nas-flexgroup`, `ontap-san`, `ontap-san-economy`
`backendName`	Nom personnalisé ou le stockage backend	Nom du pilote + "_" + dataLIF
`managementLIF`	Adresse IP d'un cluster ou d'une LIF de gestion SVM. Un nom de domaine complet (FQDN) peut être spécifié. Il est possible de configurer l'utilisation d'adresses IPv6 si Trident a été installé avec l'option IPv6. Les adresses IPv6 doivent être définies entre crochets, par exemple : [28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]. Si vous fournissez le `fsxFilesystemID` sous le champ `aws`, il n'est pas nécessaire de fournir le `managementLIF` car Trident récupère les informations SVM `managementLIF` depuis AWS. Vous devez donc fournir les identifiants d'un utilisateur sous la SVM (par exemple : vsadmin) et l'utilisateur doit avoir le rôle `vsadmin`.	"10.0.0.1", "[2001:1234:abcd::fefe]"

Paramètre	Description	Exemple
`dataLIF`	Adresse IP de l'interface logique de protocole (LIF). **Pilotes ONTAP NAS** : NetApp recommande de spécifier dataLIF. Si elle n'est pas fournie, Trident récupère les dataLIF depuis la SVM. Vous pouvez spécifier un nom de domaine complet (FQDN) à utiliser pour les opérations de montage NFS, ce qui vous permet de créer un DNS à répartition de charge (round-robin) entre plusieurs dataLIF. Peut être modifié après la configuration initiale. **Pilotes ONTAP SAN** : ne pas spécifier pour iSCSI. Trident utilise ONTAP Selective LUN Map pour découvrir les LIF iSCSI nécessaires à l'établissement d'une session multipath. Un avertissement est généré si dataLIF est explicitement défini. Il est possible de configurer l'utilisation d'adresses IPv6 si Trident a été installé avec l'option IPv6. Les adresses IPv6 doivent être définies entre crochets, par exemple : [28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555].	
`autoExportPolicy`	Activer la création et la mise à jour automatiques des règles d'export [Booléen]. En utilisant les options `autoExportPolicy` et `autoExportCIDRs`, Trident peut gérer les règles d'export automatiquement.	`false`
`autoExportCIDRs`	Liste des CIDR pour filtrer les adresses IP des nœuds Kubernetes lorsque `autoExportPolicy` est activé. En utilisant les options `autoExportPolicy` et `autoExportCIDRs`, Trident peut gérer les règles d'export automatiquement.	"["0.0.0.0/0", "::/0"]"
`labels`	Ensemble d'étiquettes arbitraires au format JSON à appliquer aux volumes	""

Paramètre	Description	Exemple
`clientCertificate`	Valeur encodée en Base64 du certificat client. Utilisé pour l'authentification par certificat	""
`clientPrivateKey`	Valeur encodée en Base64 de la clé privée du client. Utilisé pour l'authentification par certificat	""
`trustedCACertificate`	Valeur encodée en Base64 du certificat d'autorité de certification de confiance. Facultatif. Utilisé pour l'authentification par certificat.	""
`username`	Nom d'utilisateur pour se connecter au cluster ou à la SVM. Utilisé pour l'authentification par identifiants. Par exemple, vsadmin.	
`password`	Mot de passe permettant de se connecter au cluster ou à la SVM. Utilisé pour l'authentification par identifiants.	
`svm`	Machine virtuelle de stockage à utiliser	Dérivé si un LIF de gestion SVM est spécifié.
`storagePrefix`	Préfixe utilisé lors du provisionnement de nouveaux volumes dans la SVM. Ne peut pas être modifié après création. Pour mettre à jour ce paramètre, vous devrez créer un nouveau backend.	`trident`
`limitAggregateUsage`	**Ne pas spécifier pour Amazon FSx for NetApp ONTAP.** Les `fsxadmin` et `vsadmin` fournis ne contiennent pas les autorisations requises pour récupérer l'utilisation agrégée et la limiter à l'aide de Trident.	Ne pas utiliser.
`limitVolumeSize`	L'approvisionnement échoue si la taille du volume demandée dépasse cette valeur. Limite également la taille maximale des volumes qu'il gère pour les qtrees et les LUN, et l' `qtreesPerFlexvol`option permet de personnaliser le nombre maximal de qtrees par volume FlexVol	"" (non appliqué par défaut)
`lunsPerFlexvol`	Nombre maximal de LUN par volume FlexVol, doit être compris entre 50 et 200. SAN uniquement.	"100"

Paramètre	Description	Exemple
`debugTraceFlags`	Options de débogage à utiliser lors du dépannage. Exemple, {"api":false, "method":true} Ne pas utiliser `debugTraceFlags` sauf si vous effectuez un dépannage et avez besoin d'un journal détaillé.	null
`nfsMountOptions`	Liste d'options de montage NFS séparées par des virgules. Les options de montage pour les volumes persistants Kubernetes sont généralement spécifiées dans les classes de stockage, mais si aucune option de montage n'est spécifiée dans une classe de stockage, Trident utilisera les options de montage spécifiées dans le fichier de configuration du backend de stockage. Si aucune option de montage n'est spécifiée ni dans la classe de stockage ni dans le fichier de configuration, Trident n'appliquera aucune option de montage au volume persistant associé.	""
`nasType`	Configurez la création de volumes NFS ou SMB. Les options sont `nfs`, `smb` ou null. **Doit être défini sur `smb` pour les volumes SMB.** La valeur null correspond par défaut à des volumes NFS.	`nfs`
`qtreesPerFlexvol`	Nombre maximal de Qtrees par volume FlexVol, doit être compris dans la plage [50, 300]	`"200"`
`smbShare`	Vous pouvez spécifier l'un des éléments suivants : le nom d'un partage SMB créé à l'aide de la console de gestion Microsoft ou de l'interface de ligne de commande ONTAP, ou un nom permettant à Trident de créer le partage SMB. Ce paramètre est requis pour les backends Amazon FSx for ONTAP.	`smb-share`

Paramètre	Description	Exemple
`useREST`	Paramètre booléen à utiliser pour les API REST ONTAP. Lorsqu'il est défini sur `true`, Trident utilisera les API REST ONTAP pour communiquer avec le backend. Cette fonctionnalité nécessite ONTAP 9.11.1 ou une version ultérieure. De plus, le rôle de connexion ONTAP utilisé doit avoir accès à l'application `ontap`. Cela est satisfait par les rôles prédéfinis `vsadmin` et `cluster-admin`.	`false`
`aws`	Vous pouvez spécifier les éléments suivants dans le fichier de configuration pour AWS FSx for ONTAP : - `fsxFilesystemID` : Spécifiez l'ID du système de fichiers AWS FSx. - `apiRegion` : Nom de la région de l'API AWS. - `apikey` : Clé d'API AWS. - `secretKey` : Clé secrète AWS.	`""` `""` `""`
`credentials`	Spécifiez les informations d'identification de la SVM FSx à stocker dans AWS Secrets Manager. - `name` : Nom de ressource Amazon (ARN) du secret, qui contient les informations d'identification de la SVM. - `type` : Définissez sur `awsarn`. Consultez "Créer un secret AWS Secrets Manager" pour plus d'informations.	

Options de configuration backend pour le provisionnement des volumes

Vous pouvez contrôler le provisionnement par défaut à l'aide de ces options dans la section `defaults` de la configuration. Pour un exemple, consultez les exemples de configuration ci-dessous.

Paramètre	Description	Défaut
`spaceAllocation`	Allocation d'espace pour les LUNs	`true`
`spaceReserve`	Mode de réservation d'espace ; "none" (fin) ou "volume" (épais)	`none`
`snapshotPolicy`	Stratégie de snapshot à utiliser	`none`

Paramètre	Description	Défaut
`qosPolicy`	Groupe de règles QoS à attribuer aux volumes créés. Choisissez l'un des qosPolicy ou adaptiveQosPolicy par pool de stockage ou backend. L'utilisation des groupes de règles QoS avec Trident requiert ONTAP 9.8 ou une version ultérieure. Vous devez utiliser un groupe de règles QoS non partagé et vous assurer que le groupe de règles est appliqué individuellement à chaque composant. Un groupe de règles QoS partagé impose une limite au débit total de toutes les charges de travail.	""
`adaptiveQosPolicy`	Groupe de règles QoS adaptatives à attribuer aux volumes créés. Choisissez l'un des qosPolicy ou adaptiveQosPolicy par pool de stockage ou backend. Non pris en charge par ontap-nas-economy.	""
`snapshotReserve`	Pourcentage du volume réservé aux snapshots "0"	Si `snapshotPolicy` est `none`, `else`""
`splitOnClone`	Séparer un clone de son parent lors de sa création	`false`
`encryption`	Activez le chiffrement de volume NetApp (NVE) sur le nouveau volume ; la valeur par défaut est `false`. NVE doit être sous licence et activé sur le cluster pour utiliser cette option. Si NAE est activé sur le backend, tout volume provisionné dans Trident sera activé pour NAE. Pour plus d'informations, consultez : "Comment Trident fonctionne avec NVE et NAE".	`false`
`luksEncryption`	Activez le chiffrement LUKS. Consultez "Utilisez Linux Unified Key Setup (LUKS)". SAN uniquement.	""
`tieringPolicy`	Politique de hiérarchisation à utiliser `none`	
`unixPermissions`	Mode pour les nouveaux volumes. **Laisser vide pour les volumes SMB.**	""

Paramètre	Description	Défaut
`securityStyle`	Style de sécurité pour les nouveaux volumes. NFS prend en charge `mixed` et `unix` styles de sécurité. SMB prend en charge `mixed` et `ntfs` styles de sécurité.	La valeur par défaut de NFS est `unix`. La valeur par défaut de SMB est `ntfs`.

Configurer un PVC

Cette section explique comment créer un PersistentVolumeClaim (PVC) qui utilise la StorageClass Kubernetes configurée pour demander un PV. En cas de succès, vous pourrez ensuite monter le PV sur un pod.

Créer le PVC

Un "*PersistentVolumeClaim*" (PVC) est une demande d'accès au PersistentVolume sur le cluster. Le PVC peut être configuré pour demander un espace de stockage d'une certaine taille ou un certain mode d'accès. En utilisant la StorageClass associée, l'administrateur du cluster peut contrôler plus que la taille et le mode d'accès du PersistentVolume—comme les performances ou le niveau de service.

Après avoir créé le backend Trident et le StorageClass, vous pouvez créer un PVC. Après la création du PVC, vous pouvez monter le volume dans un pod.

Exemples de manifestes

Les exemples suivants illustrent les options de configuration de base du PVC.

PVC avec accès RWX

Cet exemple montre un PVC de base avec un accès RWX qui est associé à un StorageClass nommé `basic-csi`.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc-storage
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: ontap-gold
```

Exemple d'utilisation de PVC avec iSCSI

Cet exemple montre un PVC de base pour iSCSI avec accès RWO associé à un StorageClass nommé `protection-gold`.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc-san
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: protection-gold
```

Créer un PVC

Étapes

1. Créez le PVC.

```
kubectl create -f pvc.yaml
```

2. Vérifiez l'état du PVC.

```
kubectl get pvc
```

```
NAME        STATUS VOLUME     CAPACITY ACCESS MODES STORAGECLASS AGE
pvc-storage Bound  pv-name 2Gi      RWO                          5m
```

Consultez "Objets Kubernetes et Trident" pour plus de détails sur la manière dont les classes de stockage interagissent avec le `PersistentVolumeClaim` et les paramètres permettant de contrôler la façon dont Trident provisionne les volumes.

Déployer une application

Une fois la classe de stockage et le PVC créés, vous pouvez monter le PV sur un pod. Cette section présente la commande et la configuration d'exemple pour associer le PV à un pod.

Déployer une application exemple

Étapes

1. Montez le volume dans un pod.

```
kubectl create -f pv-pod.yaml
```

Ces exemples montrent des configurations de base pour attacher le PVC à un pod : **Configuration de base** :

```
kind: Pod
apiVersion: v1
metadata:
  name: pv-pod
spec:
  volumes:
    - name: pv-storage
      persistentVolumeClaim:
       claimName: basic
  containers:
    - name: pv-container
      image: nginx
      ports:
        - containerPort: 80
          name: "http-server"
      volumeMounts:
        - mountPath: "/my/mount/path"
          name: pv-storage
```

REMARQUE Vous pouvez surveiller la progression en utilisant `kubectl get pod --watch`.

2. Vérifiez que le volume est monté sur `/my/mount/path`.

```
kubectl exec -it pv-pod -- df -h /my/mount/path
```

```
Filesystem                                                      Size
Used  Avail Use%  Mounted on
192.168.188.78:/trident_pvc_ae45ed05_3ace_4e7c_9080_d2a83ae03d06  1.1G
320K  1.0G   1%   /my/mount/path
```

Vous pouvez maintenant supprimer le Pod. L'application Pod n'existera plus, mais le volume restera.

```
kubectl delete pod pv-pod
```

Configurez le module complémentaire Trident EKS sur un cluster EKS

NetApp Trident simplifie la gestion du stockage Amazon FSx for NetApp ONTAP dans Kubernetes afin de permettre à vos développeurs et administrateurs de se concentrer sur le déploiement des applications. L'add-on NetApp Trident EKS inclut les derniers

correctifs de sécurité, les corrections de bogues, et est validé par AWS pour fonctionner avec Amazon EKS. L'add-on EKS vous permet de garantir de manière cohérente que vos clusters Amazon EKS sont sécurisés et stables, et de réduire la quantité de travail nécessaire pour installer, configurer et mettre à jour les add-ons.

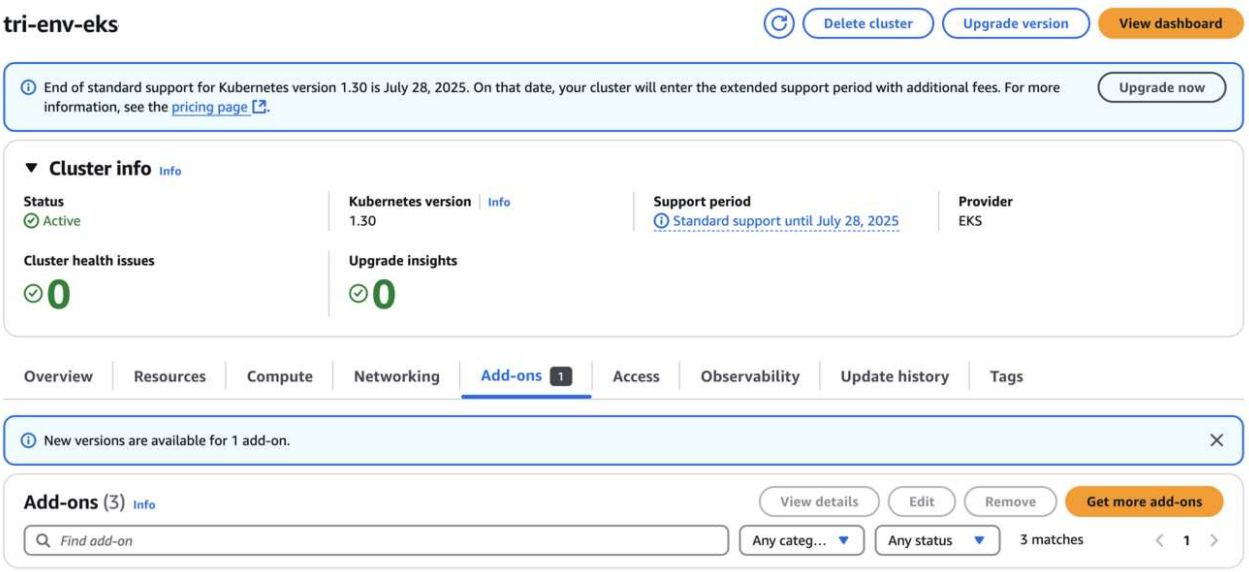
Prérequis

Assurez-vous de disposer des éléments suivants avant de configurer le module complémentaire Trident pour AWS EKS :

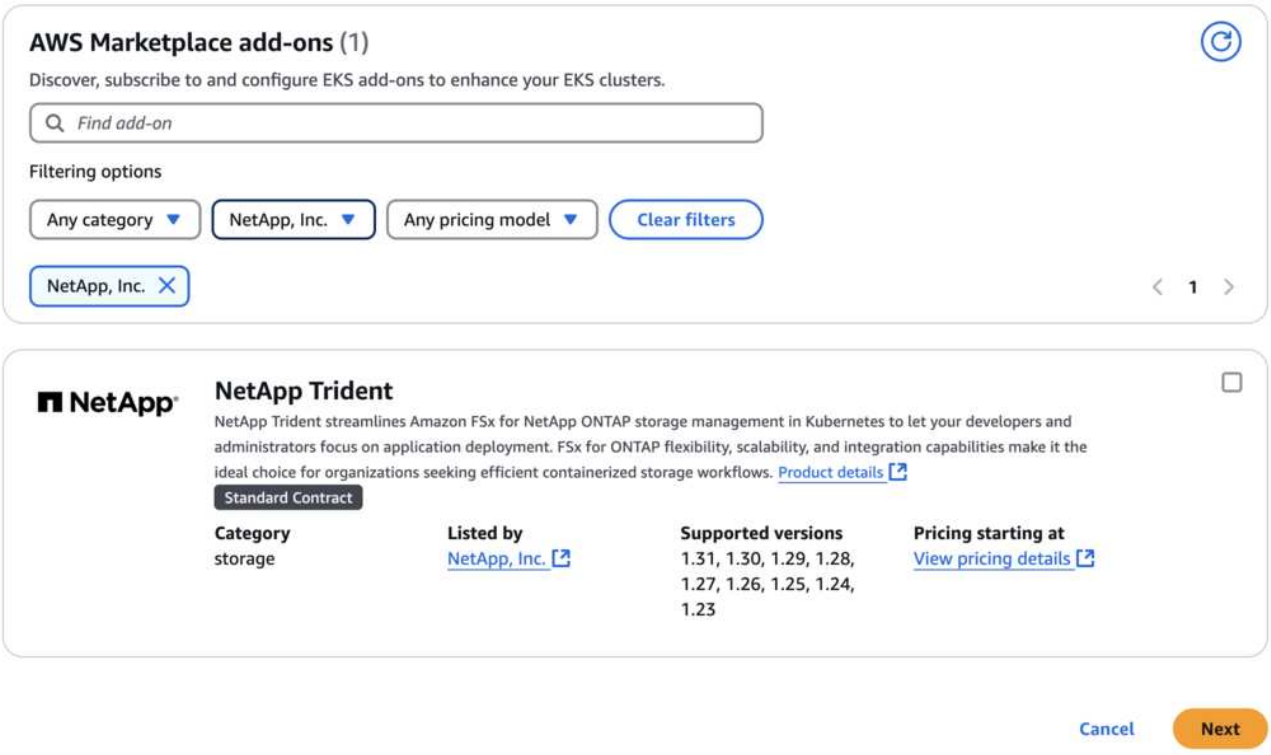
- Un compte de cluster Amazon EKS disposant des autorisations nécessaires pour gérer les modules complémentaires. Consultez "Modules complémentaires Amazon EKS".
- Autorisations AWS pour la AWS marketplace :
  `"aws-marketplace:ViewSubscriptions",`
  `"aws-marketplace:Subscribe",`
  `"aws-marketplace:Unsubscribe`
- Type d'AMI : Amazon Linux 2 (AL2_x86_64) ou Amazon Linux 2 Arm(AL2_ARM_64)
- Type de nœud : AMD ou ARM
- Un système de fichiers Amazon FSx for NetApp ONTAP existant

Étapes

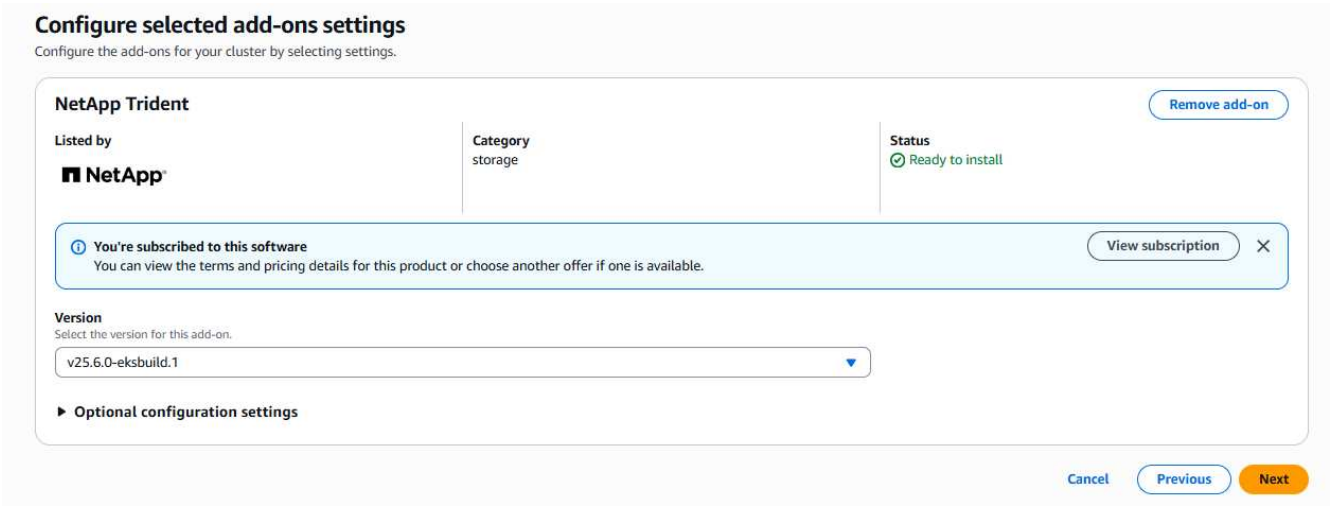
1. Veillez à créer un rôle IAM et un secret AWS pour permettre aux pods EKS d'accéder aux ressources AWS. Pour obtenir des instructions, consultez "Créez un rôle IAM et un secret AWS".
2. Sur votre cluster Kubernetes EKS, accédez à l'onglet **Add-ons**.



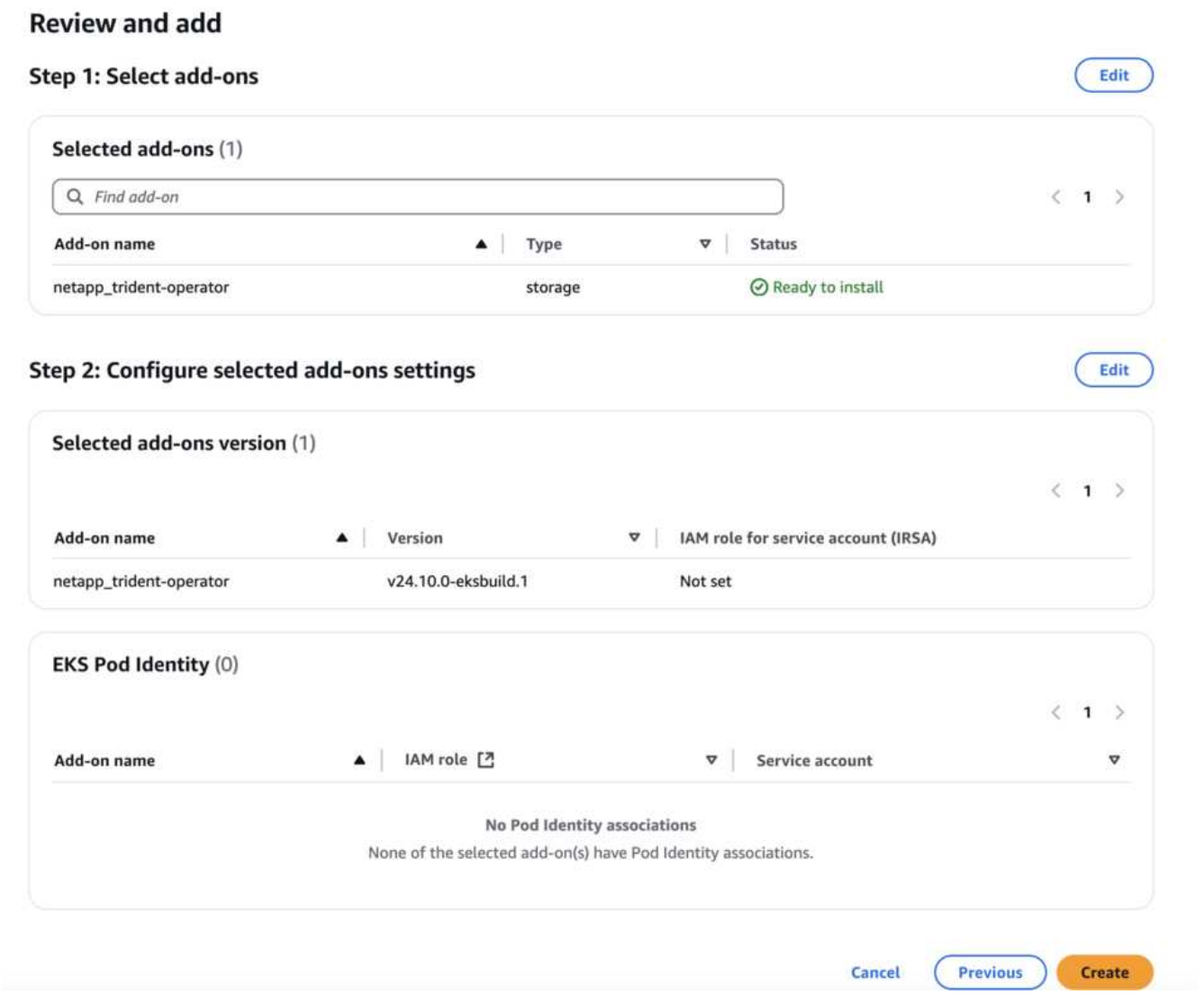
3. Accédez à **AWS Marketplace add-ons** et choisissez la catégorie *storage*.



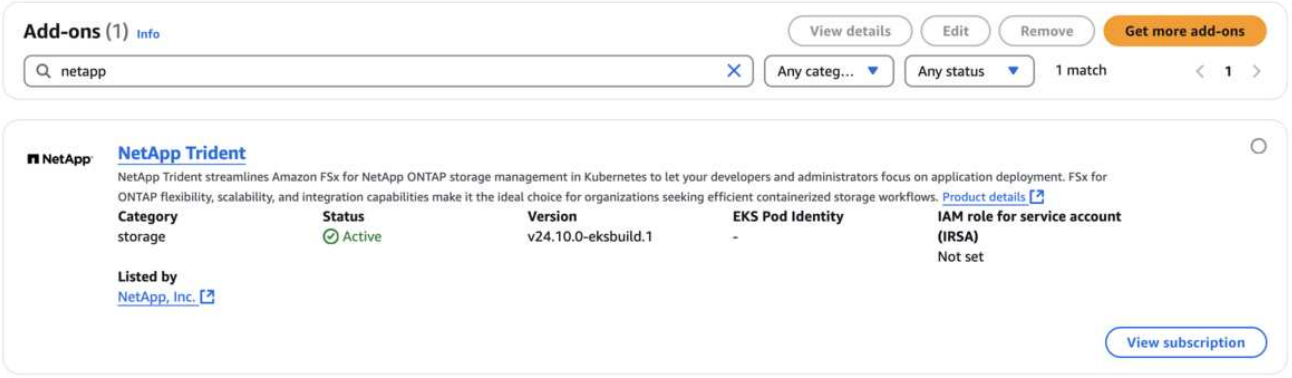
4. Repérez **NetApp Trident** et cochez la case de l'extension Trident, puis cliquez sur **Suivant**.
5. Choisissez la version souhaitée du module complémentaire.



6. Configurez les paramètres du module complémentaire requis.



7. Si vous utilisez IRSA (IAM roles for service account), reportez-vous aux étapes de configuration supplémentaires "ici".
8. Sélectionnez **Create**.
9. Vérifiez que le statut du module complémentaire est *Active*.



10. Exécutez la commande suivante pour vérifier que Trident est correctement installé sur le cluster :

```
kubectl get pods -n trident
```

11. Poursuivez l'installation et configurez le stockage backend. Pour plus d'informations, consultez "Configurer le backend de stockage".

Installez/désinstallez l'add-on Trident EKS à l'aide de la CLI

Installez le module complémentaire NetApp Trident EKS à l'aide de l'interface de ligne de commande :

La commande suivante installe le module complémentaire Trident EKS :
`eksctl create addon --cluster clusterName --name netapp_trident-operator --version v25.6.0-eksbuild.1` (avec une version dédiée)

L'exemple de commande suivant installe le Trident EKS add-on version 25.6.1 :
`eksctl create addon --cluster clusterName --name netapp_trident-operator --version v25.6.1-eksbuild.1` (avec une version dédiée)

L'exemple de commande suivant installe le Trident EKS add-on version 25.6.2 :
`eksctl create addon --cluster clusterName --name netapp_trident-operator --version v25.6.2-eksbuild.1` (avec une version dédiée)

Désinstallez le module complémentaire NetApp Trident EKS à l'aide de l'interface de ligne de commande :

La commande suivante désinstalle le module complémentaire Trident EKS :

```
eksctl delete addon --cluster K8s-arm --name netapp_trident-operator
```

Créer des backends avec kubectl

Un backend définit la relation entre Trident et un système de stockage. Il indique à Trident comment communiquer avec ce système de stockage et comment Trident doit provisionner des volumes à partir de celui-ci. Après l'installation de Trident, l'étape suivante consiste à créer un backend. La `TridentBackendConfig` Custom Resource Definition (CRD) permet de créer et de gérer des backends Trident directement via l'interface Kubernetes. Vous pouvez le faire en utilisant `kubectl` ou l'outil CLI équivalent pour votre distribution Kubernetes.

`TridentBackendConfig`

`TridentBackendConfig` (`tbc`, `tbconfig`, `tbackendconfig`) est une CRD frontale avec espace de noms qui vous permet de gérer les backends Trident à l'aide de `kubectl`. Les administrateurs Kubernetes et de stockage peuvent désormais créer et gérer des backends directement via l'interface de ligne de commande Kubernetes sans avoir besoin d'un utilitaire en ligne de commande dédié (`tridentctl`).

Lors de la création d'un `TridentBackendConfig` objet, les événements suivants se produisent :

- Un backend est créé automatiquement par Trident en fonction de la configuration que vous fournissez. Ceci est représenté en interne par une `TridentBackend` (`tbe`, `tridentbackend`) CR.
- Le `TridentBackendConfig` est lié de manière unique à un `TridentBackend` qui a été créé par

Trident.

Chacune `TridentBackendConfig` maintient une correspondance un-à-un avec une `TridentBackend`. La première est l'interface fournie à l'utilisateur pour concevoir et configurer les backends ; la seconde est la façon dont Trident représente l'objet backend réel.

AVERTISSEMENT	`TridentBackend` Les CR sont créés automatiquement par Trident. Vous **ne devez pas** les modifier. Si vous souhaitez mettre à jour les backends, faites-le en modifiant l'objet `TridentBackendConfig`.

Voir l'exemple suivant pour le format du `TridentBackendConfig` CR :

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-san
spec:
  version: 1
  backendName: ontap-san-backend
  storageDriverName: ontap-san
  managementLIF: 10.0.0.1
  dataLIF: 10.0.0.2
  svm: trident_svm
  credentials:
    name: backend-tbc-ontap-san-secret
```

Vous pouvez également consulter les exemples dans le répertoire "trident-installer" pour des configurations types pour la plateforme/le service de stockage souhaité.

Le `spec` prend des paramètres de configuration spécifiques au backend. Dans cet exemple, le backend utilise le `ontap-san` storage driver et utilise les paramètres de configuration qui sont présentés ici. Pour la liste des options de configuration pour votre storage driver souhaité, consultez le "Informations de configuration du backend pour votre pilote de stockage".

La `spec` section inclut également `credentials` et `deletionPolicy` des champs, qui sont nouvellement introduits dans le `TridentBackendConfig` CR :

- `credentials`: Ce paramètre est obligatoire et contient les identifiants utilisés pour l'authentification auprès du système/service de stockage. Ceci est défini sur un Secret Kubernetes créé par l'utilisateur. Les identifiants ne peuvent pas être transmis en clair et cela entraînera une erreur.
- `deletionPolicy` : Ce champ définit ce qui doit se produire lorsque le `TridentBackendConfig` est supprimé. Il peut prendre l'une des deux valeurs possibles :
  - `delete`: Cela entraîne la suppression de la `TridentBackendConfig` CR et du backend associé. Il s'agit de la valeur par défaut.
  - `retain` : Lorsqu'un `TridentBackendConfig` CR est supprimé, la définition du backend reste présente et peut être gérée avec `tridentctl`. La configuration de la politique de suppression sur `retain` permet aux utilisateurs de revenir à une version antérieure (pré-21.04) et de conserver les backends créés. La valeur de ce champ peut être mise à jour après qu'un `TridentBackendConfig`

est créé.

REMARQUE	Le nom d'un backend est défini à l'aide de `spec.backendName`. S'il n'est pas spécifié, le nom du backend est défini sur le nom de l'objet `TridentBackendConfig` (metadata.name). Il est recommandé de définir explicitement les noms des backends à l'aide de `spec.backendName`.
ASTUCE	Les backends qui ont été créés avec `tridentctl` n'ont pas d'objet `TridentBackendConfig` associé. Vous pouvez choisir de gérer ces backends avec `kubectl` en créant un CR `TridentBackendConfig`. Il faut veiller à spécifier des paramètres de configuration identiques (tels que `spec.backendName`, `spec.storagePrefix`, `spec.storageDriverName`, etc.). Trident associera automatiquement le nouvel objet `TridentBackendConfig` au backend préexistant.

Aperçu des étapes

Pour créer un nouveau backend en utilisant `kubectl`, vous devez procéder comme suit :

1. Créez un "Secret Kubernetes". Le secret contient les identifiants dont Trident a besoin pour communiquer avec le cluster/service de stockage.
2. Créez un `TridentBackendConfig` objet. Celui-ci contient des informations spécifiques sur le cluster/service de stockage et fait référence au secret créé à l'étape précédente.

Après avoir créé un backend, vous pouvez observer son statut en utilisant `kubectl get tbc <tbc-name> -n <trident-namespace>` et recueillir des informations supplémentaires.

Étape 1 : Créer un secret Kubernetes

Créez un secret contenant les identifiants d'accès pour le backend. Ceci est unique à chaque service/plateforme de stockage. Voici un exemple :

```
kubectl -n trident create -f backend-tbc-ontap-san-secret.yaml
```

```
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-ontap-san-secret
type: Opaque
stringData:
  username: cluster-admin
  password: password
```

Ce tableau récapitule les champs qui doivent figurer dans le Secret pour chaque plateforme de stockage :

Description des champs secrets de la plateforme de stockage	Secret	Description des champs
Azure NetApp Files	clientID	L'identifiant client issu de l'enregistrement d'une application
Element (NetApp HCI/SolidFire)	Point de terminaison	MVIP pour le cluster SolidFire avec les identifiants du locataire
ONTAP	nom d'utilisateur	Nom d'utilisateur pour se connecter au cluster/SVM. Utilisé pour l'authentification basée sur les identifiants
ONTAP	mot de passe	Mot de passe pour se connecter au cluster/SVM. Utilisé pour l'authentification basée sur les identifiants
ONTAP	clientPrivateKey	Valeur encodée en Base64 de la clé privée du client. Utilisée pour l'authentification basée sur un certificat
ONTAP	chapUsername	Nom d'utilisateur entrant. Requis si useCHAP=true. Pour `ontap-san` et `ontap-san-economy`
ONTAP	chapInitiatorSecret	Secret de l'initiateur CHAP. Requis si useCHAP=true. Pour `ontap-san` et `ontap-san-economy`
ONTAP	chapTargetUsername	Nom d'utilisateur cible. Requis si useCHAP=true. Pour `ontap-san` et `ontap-san-economy`
ONTAP	chapTargetInitiatorSecret	Secret de l'initiateur de la cible CHAP. Requis si useCHAP=true. Pour `ontap-san` et `ontap-san-economy`

Le secret créé à cette étape sera référencé dans le champ `spec.credentials` de l'objet `TridentBackendConfig` qui est créé à l'étape suivante.

Étape 2 : Créer la `TridentBackendConfig` demande de changement

Vous êtes maintenant prêt à créer votre `TridentBackendConfig` CR. Dans cet exemple, un backend qui utilise le `ontap-san` driver est créé à l'aide de l' `TridentBackendConfig` objet présenté ci-dessous :

```
kubectl -n trident create -f backend-tbc-ontap-san.yaml
```

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-san
spec:
  version: 1
  backendName: ontap-san-backend
  storageDriverName: ontap-san
  managementLIF: 10.0.0.1
  dataLIF: 10.0.0.2
  svm: trident_svm
  credentials:
    name: backend-tbc-ontap-san-secret
```

Étape 3 : Vérifiez le statut du `TridentBackendConfig` CR

Maintenant que vous avez créé le `TridentBackendConfig` CR, vous pouvez vérifier l'état. Voir l'exemple suivant :

```
kubectl -n trident get tbc backend-tbc-ontap-san
NAME                    BACKEND NAME          BACKEND UUID
PHASE   STATUS
backend-tbc-ontap-san   ontap-san-backend     8d24fce7-6f60-4d4a-8ef6-
bab2699e6ab8   Bound   Success
```

Un backend a été créé avec succès et lié au `TridentBackendConfig` CR.

La phase peut prendre l'une des valeurs suivantes :

- `Bound` : Le `TridentBackendConfig`CR est associé à un backend, et ce backend contient `configRef`défini sur l' `TridentBackendConfig`uid du CR.
- `Unbound` : Représenté à l'aide de `""`. L'objet `TridentBackendConfig` n'est pas lié à un backend. Tous les nouveaux CR créés `TridentBackendConfig` sont par défaut dans cette phase. Après le changement de phase, il ne peut plus revenir à l'état Unbound.
- `Deleting` : Le `TridentBackendConfig`CR `deletionPolicy` a été configuré pour être supprimé. Lorsque le `TridentBackendConfig`CR est supprimé, il passe à l'état Deleting.
  - Si aucune revendication de volume persistant (PVC) n'existe sur le backend, la suppression du `TridentBackendConfig` entraînera Trident à supprimer le backend ainsi que le `TridentBackendConfig` CR.
  - Si une ou plusieurs PVC sont présentes sur le backend, il passe en état de suppression. Le `TridentBackendConfig` CR entre ensuite également en phase de suppression. Le backend et

`TridentBackendConfig` ne sont supprimés qu'après la suppression de toutes les PVC.

- `Lost` : Le backend associé au `TridentBackendConfig` CR a été supprimé accidentellement ou intentionnellement et le `TridentBackendConfig` CR possède toujours une référence vers le backend supprimé. Le `TridentBackendConfig` CR peut toujours être supprimé, quelle que soit la valeur de `deletionPolicy`.
- `Unknown` : Trident ne parvient pas à déterminer l'état ou l'existence du backend associé au `TridentBackendConfig` CR. Par exemple, si le serveur API ne répond pas ou si le `tridentbackends.trident.netapp.io` CRD est manquant. Cela peut nécessiter une intervention.

À ce stade, le backend est créé avec succès ! Plusieurs opérations supplémentaires peuvent être gérées, telles que "mises à jour du backend et suppressions du backend".

(Facultatif) Étape 4 : Obtenir plus de détails

Vous pouvez exécuter la commande suivante pour obtenir plus d'informations sur votre backend :

```
kubectl -n trident get tbc backend-tbc-ontap-san -o wide
```

```
NAME                    BACKEND NAME        BACKEND UUID
PHASE   STATUS    STORAGE DRIVER   DELETION POLICY
backend-tbc-ontap-san   ontap-san-backend   8d24fce7-6f60-4d4a-8ef6-
bab2699e6ab8   Bound   Success   ontap-san        delete
```

De plus, vous pouvez également obtenir un dump YAML/JSON de `TridentBackendConfig`.

```
kubectl -n trident get tbc backend-tbc-ontap-san -o yaml
```

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  creationTimestamp: 2021-04-21T20:45:11Z
  finalizers:
    - trident.netapp.io
  generation: 1
  name: backend-tbc-ontap-san
  namespace: trident
  resourceVersion: "947143"
  uid: 35b9d777-109f-43d5-8077-c74a4559d09c
spec:
  backendName: ontap-san-backend
  credentials:
    name: backend-tbc-ontap-san-secret
  managementLIF: 10.0.0.1
  dataLIF: 10.0.0.2
  storageDriverName: ontap-san
  svm: trident_svm
  version: 1
status:
  backendInfo:
    backendName: ontap-san-backend
    backendUUID: 8d24fce7-6f60-4d4a-8ef6-bab2699e6ab8
  deletionPolicy: delete
  lastOperationStatus: Success
  message: Backend 'ontap-san-backend' created
  phase: Bound
```

`backendInfo` contient l' `backendName` et le `backendUUID` du backend qui a été créé en réponse à la `TridentBackendConfig` CR. Le `lastOperationStatus` champ représente le statut de la dernière opération de la `TridentBackendConfig` CR, qui peut être déclenchée par l'utilisateur (par exemple, si l'utilisateur a modifié quelque chose dans `spec`) ou déclenchée par Trident (par exemple, lors des redémarrages de Trident). Il peut être soit Success soit Failed. `phase` représente le statut de la relation entre la `TridentBackendConfig` CR et le backend. Dans l'exemple ci-dessus, `phase` a la valeur Bound, ce qui signifie que la `TridentBackendConfig` CR est associée au backend.

Vous pouvez exécuter la commande `kubectl -n trident describe tbc <tbc-cr-name>` pour obtenir les détails des journaux d'événements.

AVERTISSEMENT	Vous ne pouvez pas mettre à jour ou supprimer un backend qui contient un objet associé `TridentBackendConfig` en utilisant `tridentctl`. Pour comprendre les étapes nécessaires pour passer de `tridentctl` à `TridentBackendConfig`, "voir ici".

Gérer les backends

Effectuez la gestion du backend avec kubectl

Découvrez comment effectuer des opérations de gestion backend en utilisant `kubectl`.

Supprimer un backend

En supprimant un `TridentBackendConfig`, vous indiquez à Trident de supprimer ou de conserver les backends (selon `deletionPolicy`). Pour supprimer un backend, assurez-vous que `deletionPolicy` est défini sur delete. Pour supprimer uniquement le `TridentBackendConfig`, assurez-vous que `deletionPolicy` est défini sur retain. Cela garantit que le backend est toujours présent et peut être géré à l'aide de `tridentctl`.

Exécutez la commande suivante :

```
kubectl delete tbc <tbc-name> -n trident
```

Trident ne supprime pas les secrets Kubernetes qui étaient utilisés par `TridentBackendConfig`. L'utilisateur Kubernetes est responsable du nettoyage des secrets. Il faut faire attention lors de la suppression des secrets. Vous ne devez supprimer les secrets que s'ils ne sont pas utilisés par les backends.

Afficher les backends existants

Exécutez la commande suivante :

```
kubectl get tbc -n trident
```

Vous pouvez également exécuter `tridentctl get backend -n trident` ou `tridentctl get backend -o yaml -n trident` pour obtenir une liste de tous les backends existants. Cette liste inclura également les backends qui ont été créés avec `tridentctl`.

Mettre à jour un backend

Il peut exister plusieurs raisons de mettre à jour un backend :

- Les identifiants d'accès au système de stockage ont changé. Pour mettre à jour les identifiants, le Secret Kubernetes utilisé dans l' `TridentBackendConfig`objet doit être mis à jour. Trident mettra automatiquement à jour le backend avec les identifiants les plus récents fournis. Exécutez la commande suivante pour mettre à jour le Secret Kubernetes :

  ```
  kubectl apply -f <updated-secret-file.yaml> -n trident
  ```

- Les paramètres (tels que le nom de la ONTAP SVM utilisée) doivent être mis à jour.
  - Vous pouvez mettre à jour `TridentBackendConfig` les objets directement via Kubernetes à l'aide de la commande suivante :

```
kubectl apply -f <updated-backend-file.yaml>
```

- Vous pouvez également apporter des modifications à la `TridentBackendConfig`CR existante à l'aide de la commande suivante :

```
kubectl edit tbc <tbc-name> -n trident
```

REMARQUE

- En cas d'échec d'une mise à jour du backend, le backend conserve sa dernière configuration connue. Vous pouvez consulter les journaux pour en déterminer la cause en exécutant `kubectl get tbc <tbc-name> -o yaml -n trident` ou `kubectl describe tbc <tbc-name> -n trident`.
- Après avoir identifié et corrigé le problème avec le fichier de configuration, vous pouvez relancer la commande de mise à jour.

Effectuez la gestion du backend avec tridentctl

Découvrez comment effectuer des opérations de gestion backend en utilisant `tridentctl`.

Créer un backend

Après avoir créé un "fichier de configuration backend", exécutez la commande suivante :

```
tridentctl create backend -f <backend-file> -n trident
```

Si la création du backend échoue, quelque chose n'allait pas avec la configuration du backend. Vous pouvez consulter les journaux pour en déterminer la cause en exécutant la commande suivante :

```
tridentctl logs -n trident
```

Après avoir identifié et corrigé le problème avec le fichier de configuration, vous pouvez simplement exécuter la commande `create` à nouveau.

Supprimer un backend

Pour supprimer un backend de Trident, procédez comme suit :

1. Récupérer le nom du backend :

```
tridentctl get backend -n trident
```

2. Supprimez le backend :

```
tridentctl delete backend <backend-name> -n trident
```

REMARQUE	Si Trident a provisionné des volumes et des instantanés à partir de ce backend qui existent encore, la suppression du backend empêche le provisionnement de nouveaux volumes par celui-ci. Le backend continuera d'exister dans un état « Deleting ».

Afficher les backends existants

Pour afficher les backends que Trident connaît, procédez comme suit :

- Pour obtenir un résumé, exécutez la commande suivante :

```
tridentctl get backend -n trident
```

- Pour obtenir tous les détails, exécutez la commande suivante :

```
tridentctl get backend -o json -n trident
```

Mettre à jour un backend

Après avoir créé un nouveau fichier de configuration, exécutez la commande suivante :

```
tridentctl update backend <backend-name> -f <backend-file> -n trident
```

Si la mise à jour du backend échoue, cela signifie qu'il y a un problème avec la configuration du backend ou que vous avez tenté une mise à jour invalide. Vous pouvez consulter les journaux pour en déterminer la cause en exécutant la commande suivante :

```
tridentctl logs -n trident
```

Après avoir identifié et corrigé le problème avec le fichier de configuration, vous pouvez simplement exécuter la commande `update` à nouveau.

Identifiez les classes de stockage qui utilisent un backend

Voici un exemple du type de questions auxquelles vous pouvez répondre avec le JSON que `tridentctl` génère pour les objets backend. Cela utilise l' `jq` utilitaire, que vous devez installer.

```
tridentctl get backend -o json | jq '[.items[] | {backend: .name,
storageClasses: [.storage[].storageClasses]|unique}]'
```

Cela s'applique également aux backends qui ont été créés en utilisant `TridentBackendConfig`.

Passer d'une option de gestion du backend à une autre

Découvrez les différentes manières de gérer les backends dans Trident.

Options pour la gestion des backends

Avec l'introduction de `TridentBackendConfig`, les administrateurs disposent désormais de deux méthodes distinctes pour gérer les backends. Cela soulève les questions suivantes :

- Les backends créés à l'aide de `tridentctl` peuvent-ils être gérés avec `TridentBackendConfig` ?
- Les backends créés à l'aide de `TridentBackendConfig` peuvent-ils être gérés à l'aide de `tridentctl` ?

Gérer `tridentctl` les backends à l'aide de `TridentBackendConfig`

Cette section couvre les étapes nécessaires pour gérer les backends qui ont été créés à l'aide de `tridentctl` directement via l'interface Kubernetes en créant des `TridentBackendConfig` objets.

Cela s'appliquera aux scénarios suivants :

- Les backends préexistants, qui n'ont pas de `TridentBackendConfig` parce qu'ils ont été créés avec `tridentctl`.
- De nouveaux backends ont été créés avec `tridentctl`, tandis que d'autres `TridentBackendConfig` objets existent.

Dans les deux cas, les backends resteront présents, Trident planifiant les volumes et opérant dessus. Les administrateurs ont alors deux choix :

- Continuez à utiliser `tridentctl` pour gérer les backends qui ont été créés avec.
- Liez les backends créés à l'aide de `tridentctl` à un nouvel objet `TridentBackendConfig`. Cela signifie que les backends seront gérés à l'aide de `kubectl` et non `tridentctl`.

Pour gérer un backend préexistant à l'aide de `kubectl`, vous devrez créer un `TridentBackendConfig` qui se lie au backend existant. Voici un aperçu de la façon dont cela fonctionne :

1. Créez un secret Kubernetes. Le secret contient les informations d'identification dont Trident a besoin pour communiquer avec le cluster/service de stockage.
2. Créez un `TridentBackendConfig` objet. Celui-ci contient des informations spécifiques sur le cluster/service de stockage et fait référence au secret créé à l'étape précédente. Veillez à spécifier des paramètres de configuration identiques (tels que `spec.backendName`, `spec.storagePrefix`, `spec.storageDriverName`, etc.). `spec.backendName` doit être défini sur le nom du backend existant.

Étape 0 : Identifier le backend

Pour créer une `TridentBackendConfig` qui se lie à un backend existant, vous devrez obtenir la configuration du backend. Dans cet exemple, supposons qu'un backend ait été créé à l'aide de la définition JSON suivante :

```
tridentctl get backend ontap-nas-backend -n trident
+---------------------+----------------
+--------------------------------------+--------+---------+
|          NAME       | STORAGE DRIVER |                 UUID
| STATE  | VOLUMES |
+---------------------+----------------
+--------------------------------------+--------+---------+
| ontap-nas-backend   | ontap-nas      | 52f2eb10-e4c6-4160-99fc-
96b3be5ab5d7 | online |      25 |
+---------------------+----------------
+--------------------------------------+--------+---------+
```

```
cat ontap-nas-backend.json
```

```
{
  "version": 1,
  "storageDriverName": "ontap-nas",
  "managementLIF": "10.10.10.1",
  "dataLIF": "10.10.10.2",
  "backendName": "ontap-nas-backend",
  "svm": "trident_svm",
  "username": "cluster-admin",
  "password": "admin-password",
  "defaults": {
    "spaceReserve": "none",
    "encryption": "false"
  },
  "labels": {
    "store": "nas_store"
  },
  "region": "us_east_1",
  "storage": [
    {
      "labels": {
        "app": "msoffice",
        "cost": "100"
      },
      "zone": "us_east_1a",
      "defaults": {
        "spaceReserve": "volume",
        "encryption": "true",
        "unixPermissions": "0755"
      }
    },
    {
      "labels": {
        "app": "mysqldb",
        "cost": "25"
      },
      "zone": "us_east_1d",
      "defaults": {
        "spaceReserve": "volume",
        "encryption": "false",
        "unixPermissions": "0775"
      }
    }
  ]
}
```

Étape 1 : Créer un secret Kubernetes

Créez un Secret qui contient les identifiants pour le backend, comme indiqué dans cet exemple :

```
cat tbc-ontap-nas-backend-secret.yaml
```

```
apiVersion: v1
kind: Secret
metadata:
  name: ontap-nas-backend-secret
type: Opaque
stringData:
  username: cluster-admin
  password: admin-password
```

```
kubectl create -f tbc-ontap-nas-backend-secret.yaml -n trident
secret/backend-tbc-ontap-san-secret created
```

Étape 2 : Créer un `TridentBackendConfig` CR

L'étape suivante consiste à créer un `TridentBackendConfig` CR qui se liera automatiquement à la `ontap-nas-backend` préexistante (comme dans cet exemple). Assurez-vous que les conditions suivantes sont remplies :

- Le même nom de backend est défini dans `spec.backendName`.
- Les paramètres de configuration sont identiques à ceux du backend d'origine.
- Les pools virtuels (le cas échéant) doivent conserver le même ordre que dans le backend d'origine.
- Les informations d'identification sont fournies via un Secret Kubernetes et non en texte clair.

Dans ce cas, le `TridentBackendConfig` ressemblera à ceci :

```
cat backend-tbc-ontap-nas.yaml
```

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: tbc-ontap-nas-backend
spec:
  version: 1
  storageDriverName: ontap-nas
  managementLIF: 10.10.10.1
  dataLIF: 10.10.10.2
  backendName: ontap-nas-backend
  svm: trident_svm
  credentials:
    name: mysecret
  defaults:
    spaceReserve: none
    encryption: 'false'
  labels:
    store: nas_store
  region: us_east_1
  storage:
  - labels:
      app: msoffice
      cost: '100'
    zone: us_east_1a
    defaults:
      spaceReserve: volume
      encryption: 'true'
      unixPermissions: '0755'
  - labels:
      app: mysqldb
      cost: '25'
    zone: us_east_1d
    defaults:
      spaceReserve: volume
      encryption: 'false'
      unixPermissions: '0775'
```

```
kubectl create -f backend-tbc-ontap-nas.yaml -n trident
tridentbackendconfig.trident.netapp.io/tbc-ontap-nas-backend created
```

Étape 3 : Vérifiez le statut du `TridentBackendConfig` CR

Après la `TridentBackendConfig` création, sa phase doit être `Bound`. Il doit également refléter le même nom de backend et le même UUID que le backend existant.

```
kubectl get tbc tbc-ontap-nas-backend -n trident
NAME                   BACKEND NAME          BACKEND UUID
PHASE   STATUS
tbc-ontap-nas-backend  ontap-nas-backend     52f2eb10-e4c6-4160-99fc-
96b3be5ab5d7   Bound   Success

#confirm that no new backends were created (i.e., TridentBackendConfig did
not end up creating a new backend)
tridentctl get backend -n trident
+---------------------+----------------
+--------------------------------------+--------+---------+
|          NAME       | STORAGE DRIVER |                 UUID
| STATE  | VOLUMES |
+---------------------+----------------
+--------------------------------------+--------+---------+
| ontap-nas-backend   | ontap-nas      | 52f2eb10-e4c6-4160-99fc-
96b3be5ab5d7 | online |      25 |
+---------------------+----------------
+--------------------------------------+--------+---------+
```

Le backend sera désormais entièrement géré à l'aide de l' `tbc-ontap-nas-backend` `TridentBackendConfig`objet.

Gérer `TridentBackendConfig` les backends à l'aide de `tridentctl`

```
`tridentctl` peut être utilisé pour lister les backends qui ont été créés
à l'aide de  `TridentBackendConfig`. De plus, les administrateurs peuvent
également choisir de gérer entièrement ces backends via  `tridentctl` en
supprimant  `TridentBackendConfig` et en s'assurant que
`spec.deletionPolicy` est défini sur  `retain`.
```

Étape 0 : Identifier le backend

Par exemple, supposons que le backend suivant ait été créé à l'aide de `TridentBackendConfig`:

```
kubectl get tbc backend-tbc-ontap-san -n trident -o wide
NAME                    BACKEND NAME        BACKEND UUID
PHASE   STATUS    STORAGE DRIVER   DELETION POLICY
backend-tbc-ontap-san   ontap-san-backend   81abcb27-ea63-49bb-b606-
0a5315ac5f82   Bound   Success   ontap-san        delete

tridentctl get backend ontap-san-backend -n trident
+-------------------+----------------
+--------------------------------------+--------+---------+
|       NAME        | STORAGE DRIVER |                 UUID
| STATE  | VOLUMES |
+-------------------+----------------
+--------------------------------------+--------+---------+
| ontap-san-backend | ontap-san      | 81abcb27-ea63-49bb-b606-
0a5315ac5f82 | online |      33 |
+-------------------+----------------
+--------------------------------------+--------+---------+
```

D'après le résultat, on constate que `TridentBackendConfig` a été créé avec succès et est lié à un backend [observer l'UUID du backend].

Étape 1 : Confirmer `deletionPolicy`est défini sur `retain`

Examinons la valeur de `deletionPolicy`. Celle-ci doit être définie sur `retain`. Cela garantit que lorsqu'une `TridentBackendConfig` CR est supprimée, la définition backend sera toujours présente et pourra être gérée avec `tridentctl`.

```
kubectl get tbc backend-tbc-ontap-san -n trident -o wide
NAME                    BACKEND NAME        BACKEND UUID
PHASE   STATUS    STORAGE DRIVER   DELETION POLICY
backend-tbc-ontap-san   ontap-san-backend   81abcb27-ea63-49bb-b606-
0a5315ac5f82   Bound   Success   ontap-san        delete

# Patch value of deletionPolicy to retain
kubectl patch tbc backend-tbc-ontap-san --type=merge -p
'{"spec":{"deletionPolicy":"retain"}}' -n trident
tridentbackendconfig.trident.netapp.io/backend-tbc-ontap-san patched

#Confirm the value of deletionPolicy
kubectl get tbc backend-tbc-ontap-san -n trident -o wide
NAME                    BACKEND NAME        BACKEND UUID
PHASE   STATUS    STORAGE DRIVER   DELETION POLICY
backend-tbc-ontap-san   ontap-san-backend   81abcb27-ea63-49bb-b606-
0a5315ac5f82   Bound   Success   ontap-san        retain
```

REMARQUE	Ne passez pas à l'étape suivante à moins que `deletionPolicy` soit défini sur `retain`.

Étape 2 : Supprimer le `TridentBackendConfig` CR

La dernière étape consiste à supprimer le `TridentBackendConfig` CR. Après avoir confirmé que le `deletionPolicy` est défini sur `retain`, vous pouvez procéder à la suppression :

```
kubectl delete tbc backend-tbc-ontap-san -n trident
tridentbackendconfig.trident.netapp.io "backend-tbc-ontap-san" deleted

tridentctl get backend ontap-san-backend -n trident
+-------------------+----------------
+--------------------------------------+--------+---------+
|       NAME        | STORAGE DRIVER |                 UUID
| STATE  | VOLUMES |
+-------------------+----------------
+--------------------------------------+--------+---------+
| ontap-san-backend | ontap-san      | 81abcb27-ea63-49bb-b606-
0a5315ac5f82 | online |      33 |
+-------------------+----------------
+--------------------------------------+--------+---------+
```

Lors de la suppression de l `TridentBackendConfig`objet, Trident le supprime simplement sans supprimer le backend lui-même.

Créer et gérer des classes de stockage

Créer une classe de stockage

Configurez un objet Kubernetes StorageClass et créez la classe de stockage pour indiquer à Trident comment provisionner les volumes.

Configurer un objet StorageClass Kubernetes

Le "Objet Kubernetes StorageClass" identifie Trident comme le provisionneur utilisé pour cette classe et indique à Trident comment provisionner un volume. Par exemple :

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-gold
provisioner: csi.trident.netapp.io
mountOptions:
  - nfsvers=3
  - nolock
parameters:
  backendType: "ontap-nas"
  media: "ssd"
allowVolumeExpansion: true
volumeBindingMode: Immediate
```

Consultez "Objets Kubernetes et Trident" pour plus de détails sur la manière dont les classes de stockage interagissent avec le `PersistentVolumeClaim` et les paramètres permettant de contrôler la façon dont Trident provisionne les volumes.

Créer une classe de stockage

Après avoir créé l'objet StorageClass, vous pouvez créer la classe de stockage. Exemples de classe de stockage fournit quelques exemples de base que vous pouvez utiliser ou modifier.

Étapes

1. Il s'agit d'un objet Kubernetes, utilisez donc `kubectl` pour le créer dans Kubernetes.

```
kubectl create -f sample-input/storage-class-basic-csi.yaml
```

2. Vous devriez maintenant voir une classe de stockage **basic-csi** dans Kubernetes et Trident, et Trident devrait avoir découvert les pools sur le backend.

```
kubectl get sc basic-csi
```

```
NAME         PROVISIONER             AGE
basic-csi    csi.trident.netapp.io   15h
```

```
./tridentctl -n trident get storageclass basic-csi -o json
```

```
{
  "items": [
    {
      "Config": {
        "version": "1",
        "name": "basic-csi",
        "attributes": {
          "backendType": "ontap-nas"
        },
        "storagePools": null,
        "additionalStoragePools": null
      },
      "storage": {
        "ontapnas_10.0.0.1": [
          "aggr1",
          "aggr2",
          "aggr3",
          "aggr4"
        ]
      }
    }
  ]
}
```

Exemples de classe de stockage

Trident fournit "Définitions simples de classes de stockage pour des backends spécifiques".

Vous pouvez également modifier `sample-input/storage-class-csi.yaml.templ` fichier fourni avec le programme d'installation et remplacer *`BACKEND_TYPE`* par le nom du pilote de stockage.

```
./tridentctl -n trident get backend
+-------------+---------------+--------------------------------------
+--------+---------+
|    NAME     | STORAGE DRIVER |                 UUID                 |
STATE  | VOLUMES |
+-------------+---------------+--------------------------------------
+--------+---------+
| nas-backend | ontap-nas      | 98e19b74-aec7-4a3d-8dcf-128e5033b214 |
online |       0 |
+-------------+---------------+--------------------------------------
+--------+---------+

cp sample-input/storage-class-csi.yaml.templ sample-input/storage-class-
basic-csi.yaml

# Modify __BACKEND_TYPE__ with the storage driver field above (e.g.,
ontap-nas)
vi sample-input/storage-class-basic-csi.yaml
```

Gérer les classes de stockage

Vous pouvez consulter les classes de stockage existantes, définir une classe de stockage par défaut, identifier le backend de la classe de stockage et supprimer des classes de stockage.

Afficher les classes de stockage existantes

- Pour afficher les classes de stockage Kubernetes existantes, exécutez la commande suivante :

```
kubectl get storageclass
```

- Pour afficher les détails d'une classe de stockage Kubernetes, exécutez la commande suivante :

```
kubectl get storageclass <storage-class> -o json
```

- Pour afficher les classes de stockage synchronisées de Trident, exécutez la commande suivante :

```
tridentctl get storageclass
```

- Pour afficher les détails de la classe de stockage synchronisée de Trident, exécutez la commande suivante :

```
tridentctl get storageclass <storage-class> -o json
```

Définir une classe de stockage par défaut

Kubernetes 1.6 a introduit la possibilité de définir une classe de stockage par défaut. Il s'agit de la classe de stockage qui sera utilisée pour provisionner un volume persistant si un utilisateur n'en spécifie pas dans une revendication de volume persistant (PVC).

- Définissez une classe de stockage par défaut en définissant l'annotation `storageclass.kubernetes.io/is-default-class` sur true dans la définition de la classe de stockage. Conformément à la spécification, toute autre valeur ou l'absence de l'annotation est interprétée comme false.
- Vous pouvez configurer une classe de stockage existante comme classe de stockage par défaut à l'aide de la commande suivante :

```
kubectl patch storageclass <storage-class-name> -p '{"metadata":
{"annotations":{"storageclass.kubernetes.io/is-default-class":"true"}}}'
```

- De même, vous pouvez supprimer l'annotation de classe de stockage par défaut en utilisant la commande suivante :

```
kubectl patch storageclass <storage-class-name> -p '{"metadata":
{"annotations":{"storageclass.kubernetes.io/is-default-class":"false"}}}'
```

Il existe également des exemples dans le bundle d'installation de Trident qui incluent cette annotation.

REMARQUE	Il ne devrait y avoir qu'une seule classe de stockage par défaut dans votre cluster à la fois. Kubernetes ne vous empêche pas techniquement d'en avoir plusieurs, mais il se comportera comme s'il n'y avait aucune classe de stockage par défaut.

Identifier le backend d'une classe de stockage

Voici un exemple du type de questions auxquelles vous pouvez répondre avec le JSON que `tridentctl` génère pour les objets backend Trident. Cela utilise l' `jq` utilitaire, que vous devrez peut-être installer d'abord.

```
tridentctl get storageclass -o json | jq  '[.items[] | {storageClass:
.Config.name, backends: [.storage]|unique}]'
```

Supprimer une classe de stockage

Pour supprimer une classe de stockage de Kubernetes, exécutez la commande suivante :

```
kubectl delete storageclass <storage-class>
```

`<storage-class>` devrait être remplacé par votre classe de stockage.

Tous les volumes persistants qui ont été créés via cette classe de stockage resteront intacts, et Trident continuera de les gérer.

REMARQUE	Trident impose un espace vide `fsType` pour les volumes qu'il crée. Pour les backends iSCSI, il est recommandé de l'imposer `parameters.fsType` dans le StorageClass. Vous devez supprimer les StorageClasses existants et les recréer avec `parameters.fsType` spécifié.

Provisionner et gérer les volumes

Provisionner un volume

Créez un PersistentVolumeClaim (PVC) qui utilise le StorageClass Kubernetes configuré pour demander l'accès au PV. Vous pouvez ensuite monter le PV sur un pod.

Aperçu

Un "*PersistentVolumeClaim*" (PVC) est une demande d'accès au PersistentVolume sur le cluster.

Le PVC peut être configuré pour demander un espace de stockage d'une certaine taille ou un certain mode d'accès. En utilisant la StorageClass associée, l'administrateur du cluster peut contrôler plus que la taille et le mode d'accès du PersistentVolume—comme les performances ou le niveau de service.

Après avoir créé le PVC, vous pouvez monter le volume dans un pod.

Créer le PVC

Étapes

1. Créez le PVC.

```
kubectl create -f pvc.yaml
```

2. Vérifiez l'état du PVC.

```
kubectl get pvc
```

```
NAME          STATUS  VOLUME     CAPACITY   ACCESS MODES  STORAGECLASS AGE
pvc-storage Bound   pv-name    1Gi        RWO                 5m
```

1. Montez le volume dans un pod.

```
kubectl create -f pv-pod.yaml
```

REMARQUE Vous pouvez surveiller la progression en utilisant `kubectl get pod --watch`.

2. Vérifiez que le volume est monté sur `/my/mount/path`.

```
kubectl exec -it task-pv-pod -- df -h /my/mount/path
```

3. Vous pouvez maintenant supprimer le Pod. L'application Pod n'existera plus, mais le volume restera.

```
kubectl delete pod pv-pod
```

Exemples de manifestes

Exemples de manifestes PersistentVolumeClaim

Ces exemples montrent les options de configuration de base des PVC.

PVC avec accès RWO

Cet exemple montre un PVC de base avec un accès RWO qui est associé à un StorageClass nommé `basic-csi`.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc-storage
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: basic-csi
```

PVC avec NVMe/TCP

Cet exemple montre un PVC de base pour NVMe/TCP avec accès RWO qui est associé à un StorageClass nommé `protection-gold`.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
name: pvc-san-nvme
spec:
accessModes:
  - ReadWriteOnce
resources:
  requests:
    storage: 300Mi
storageClassName: protection-gold
```

Exemples de manifestes Pod

Ces exemples montrent des configurations de base pour attacher le PVC à un pod.

Configuration de base

```
kind: Pod
apiVersion: v1
metadata:
  name: pv-pod
spec:
  volumes:
    - name: storage
      persistentVolumeClaim:
       claimName: pvc-storage
  containers:
    - name: pv-container
      image: nginx
      ports:
        - containerPort: 80
          name: "http-server"
      volumeMounts:
        - mountPath: "/my/mount/path"
          name: storage
```

Configuration NVMe/TCP de base

```
apiVersion: v1
kind: Pod
metadata:
  name: pod-nginx
spec:
  volumes:
    - name: basic-pvc
      persistentVolumeClaim:
        claimName: pvc-san-nvme
  containers:
    - name: task-pv-container
      image: nginx
      volumeMounts:
        - mountPath: "/my/mount/path"
          name: basic-pvc
```

Consultez "Objets Kubernetes et Trident" pour plus de détails sur la manière dont les classes de stockage interagissent avec le `PersistentVolumeClaim` et les paramètres permettant de contrôler la façon dont Trident provisionne les volumes.

Étendre les volumes

Trident permet aux utilisateurs de Kubernetes d'étendre leurs volumes après leur création. Consultez les informations relatives aux configurations requises pour étendre les volumes iSCSI, NFS, SMB, NVMe/TCP et FC.

Étendre un volume iSCSI

Vous pouvez étendre un volume persistant iSCSI (PV) à l'aide du provisionneur CSI.

REMARQUE	L'extension de volume iSCSI est prise en charge par les `ontap-san`, `ontap-san-economy`, `solidfire-san` pilotes et nécessite Kubernetes 1.16 et versions ultérieures.

Étape 1 : Configurer le StorageClass pour prendre en charge l'extension de volume

Modifiez la définition StorageClass pour définir le champ `allowVolumeExpansion` sur `true`.

```
cat storageclass-ontapsan.yaml
```

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-san
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
allowVolumeExpansion: True
```

Pour un StorageClass déjà existant, modifiez-le pour inclure le `allowVolumeExpansion` paramètre.

Étape 2 : Créez un PVC avec le StorageClass que vous avez créé

Modifiez la définition du PVC et mettez à jour le `spec.resources.requests.storage` pour refléter la nouvelle taille souhaitée, qui doit être supérieure à la taille d'origine.

```
cat pvc-ontapsan.yaml
```

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: san-pvc
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: ontap-san
```

Trident crée un volume persistant (PV) et l'associe à cette Persistent Volume Claim (PVC).

```
kubectl get pvc
NAME      STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS   AGE
san-pvc   Bound    pvc-8a814d62-bd58-4253-b0d1-82f2885db671   1Gi
RWO            ontap-san      8s

kubectl get pv
NAME                                       CAPACITY   ACCESS MODES
RECLAIM POLICY   STATUS   CLAIM              STORAGECLASS   REASON   AGE
pvc-8a814d62-bd58-4253-b0d1-82f2885db671   1Gi        RWO
Delete           Bound    default/san-pvc    ontap-san               10s
```

Étape 3 : Définir un pod auquel se fixe le PVC

Fixez le PV à un pod pour qu'il soit redimensionné. Il existe deux scénarios lors du redimensionnement d'un PV iSCSI :

- Si le PV est attaché à un pod, Trident étend le volume sur le backend de stockage, rescanne le périphérique et redimensionne le système de fichiers.
- Lors d'une tentative de redimensionnement d'un PV non attaché, Trident étend le volume sur le stockage backend. Après que le PVC est lié à un pod, Trident rescanne le périphérique et redimensionne le système de fichiers. Kubernetes met ensuite à jour la taille du PVC après que l'opération d'extension a été effectuée avec succès.

Dans cet exemple, un pod est créé qui utilise le `san-pvc`.

```
 kubectl get pod
NAME         READY   STATUS    RESTARTS   AGE
ubuntu-pod   1/1     Running   0          65s

 kubectl describe pvc san-pvc
Name:          san-pvc
Namespace:     default
StorageClass:  ontap-san
Status:        Bound
Volume:        pvc-8a814d62-bd58-4253-b0d1-82f2885db671
Labels:        <none>
Annotations:   pv.kubernetes.io/bind-completed: yes
               pv.kubernetes.io/bound-by-controller: yes
               volume.beta.kubernetes.io/storage-provisioner:
csi.trident.netapp.io
Finalizers:    [kubernetes.io/pvc-protection]
Capacity:      1Gi
Access Modes:  RWO
VolumeMode:    Filesystem
Mounted By:    ubuntu-pod
```

Étape 4 : Développer le PV

Pour redimensionner le PV créé de 1Gi à 2Gi, modifiez la définition du PVC et mettez à jour le `spec.resources.requests.storage` à 2Gi.

```
kubectl edit pvc san-pvc
```

```
# Please edit the object below. Lines beginning with a '#' will be
ignored,
# and an empty file will abort the edit. If an error occurs while saving
this file will be
# reopened with the relevant failures.
#
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    pv.kubernetes.io/bind-completed: "yes"
    pv.kubernetes.io/bound-by-controller: "yes"
    volume.beta.kubernetes.io/storage-provisioner: csi.trident.netapp.io
  creationTimestamp: "2019-10-10T17:32:29Z"
  finalizers:
  - kubernetes.io/pvc-protection
  name: san-pvc
  namespace: default
  resourceVersion: "16609"
  selfLink: /api/v1/namespaces/default/persistentvolumeclaims/san-pvc
  uid: 8a814d62-bd58-4253-b0d1-82f2885db671
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 2Gi
 # ...
```

Étape 5 : Valider l'expansion

Vous pouvez vérifier que l'extension a fonctionné correctement en contrôlant la taille du PVC, du PV et du volume Trident :

```
kubectl get pvc san-pvc
NAME      STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS   AGE
san-pvc   Bound    pvc-8a814d62-bd58-4253-b0d1-82f2885db671   2Gi
RWO            ontap-san      11m
kubectl get pv
NAME                                       CAPACITY   ACCESS MODES
RECLAIM POLICY   STATUS   CLAIM             STORAGECLASS   REASON   AGE
pvc-8a814d62-bd58-4253-b0d1-82f2885db671   2Gi        RWO
Delete           Bound    default/san-pvc   ontap-san               12m
tridentctl get volumes -n trident
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
|                   NAME                  |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID              | STATE  | MANAGED |
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
| pvc-8a814d62-bd58-4253-b0d1-82f2885db671 | 2.0 GiB | ontap-san     |
block    | a9b7bfff-0505-4e31-b6c5-59f492e02d33 | online | true    |
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
```

Étendre un volume FC

Vous pouvez étendre un volume persistant (PV) FC en utilisant le provisionneur CSI.

REMARQUE	L'extension de volume FC est prise en charge par le `ontap-san` driver et nécessite Kubernetes 1.16 et versions ultérieures.

Étape 1 : Configurer le StorageClass pour prendre en charge l'extension de volume

Modifiez la définition StorageClass pour définir le champ `allowVolumeExpansion` sur `true`.

```
cat storageclass-ontapsan.yaml
```

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-san
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
allowVolumeExpansion: True
```

Pour un StorageClass déjà existant, modifiez-le pour inclure le `allowVolumeExpansion` paramètre.

Étape 2 : Créez un PVC avec le StorageClass que vous avez créé

Modifiez la définition du PVC et mettez à jour le `spec.resources.requests.storage` pour refléter la nouvelle taille souhaitée, qui doit être supérieure à la taille d'origine.

```
cat pvc-ontapsan.yaml
```

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: san-pvc
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: ontap-san
```

Trident crée un volume persistant (PV) et l'associe à cette Persistent Volume Claim (PVC).

```
kubectl get pvc
NAME      STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS   AGE
san-pvc   Bound    pvc-8a814d62-bd58-4253-b0d1-82f2885db671   1Gi
RWO            ontap-san      8s

kubectl get pv
NAME                                       CAPACITY   ACCESS MODES
RECLAIM POLICY   STATUS   CLAIM              STORAGECLASS   REASON   AGE
pvc-8a814d62-bd58-4253-b0d1-82f2885db671   1Gi        RWO
Delete           Bound    default/san-pvc    ontap-san               10s
```

Étape 3 : Définir un pod auquel se fixe le PVC

Fixez le PV à un pod pour qu'il soit redimensionné. Il existe deux scénarios lors du redimensionnement d'un PV FC :

- Si le PV est attaché à un pod, Trident étend le volume sur le backend de stockage, rescanne le périphérique et redimensionne le système de fichiers.
- Lors d'une tentative de redimensionnement d'un PV non attaché, Trident étend le volume sur le stockage backend. Après que le PVC est lié à un pod, Trident rescanne le périphérique et redimensionne le système de fichiers. Kubernetes met ensuite à jour la taille du PVC après que l'opération d'extension a été effectuée

avec succès.

Dans cet exemple, un pod est créé qui utilise le `san-pvc`.

```
 kubectl get pod
NAME         READY   STATUS    RESTARTS   AGE
ubuntu-pod   1/1     Running   0          65s

 kubectl describe pvc san-pvc
Name:          san-pvc
Namespace:     default
StorageClass:  ontap-san
Status:        Bound
Volume:        pvc-8a814d62-bd58-4253-b0d1-82f2885db671
Labels:        <none>
Annotations:   pv.kubernetes.io/bind-completed: yes
               pv.kubernetes.io/bound-by-controller: yes
               volume.beta.kubernetes.io/storage-provisioner:
csi.trident.netapp.io
Finalizers:    [kubernetes.io/pvc-protection]
Capacity:      1Gi
Access Modes:  RWO
VolumeMode:    Filesystem
Mounted By:    ubuntu-pod
```

Étape 4 : Développer le PV

Pour redimensionner le PV créé de 1Gi à 2Gi, modifiez la définition du PVC et mettez à jour le `spec.resources.requests.storage` à 2Gi.

```
kubectl edit pvc san-pvc
```

```
# Please edit the object below. Lines beginning with a '#' will be
ignored,
# and an empty file will abort the edit. If an error occurs while saving
this file will be
# reopened with the relevant failures.
#
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    pv.kubernetes.io/bind-completed: "yes"
    pv.kubernetes.io/bound-by-controller: "yes"
    volume.beta.kubernetes.io/storage-provisioner: csi.trident.netapp.io
  creationTimestamp: "2019-10-10T17:32:29Z"
  finalizers:
  - kubernetes.io/pvc-protection
  name: san-pvc
  namespace: default
  resourceVersion: "16609"
  selfLink: /api/v1/namespaces/default/persistentvolumeclaims/san-pvc
  uid: 8a814d62-bd58-4253-b0d1-82f2885db671
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 2Gi
 # ...
```

Étape 5 : Valider l'expansion

Vous pouvez vérifier que l'extension a fonctionné correctement en contrôlant la taille du PVC, du PV et du volume Trident :

```
kubectl get pvc san-pvc
NAME      STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS   AGE
san-pvc   Bound    pvc-8a814d62-bd58-4253-b0d1-82f2885db671   2Gi
RWO            ontap-san      11m
kubectl get pv
NAME                                       CAPACITY   ACCESS MODES
RECLAIM POLICY   STATUS   CLAIM             STORAGECLASS   REASON   AGE
pvc-8a814d62-bd58-4253-b0d1-82f2885db671   2Gi        RWO
Delete           Bound    default/san-pvc   ontap-san               12m
tridentctl get volumes -n trident
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
|                   NAME                  |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
| pvc-8a814d62-bd58-4253-b0d1-82f2885db671 | 2.0 GiB | ontap-san     |
block    | a9b7bfff-0505-4e31-b6c5-59f492e02d33 | online | true    |
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
```

Étendre un volume NFS

Trident prend en charge l'extension de volume pour les PV NFS provisionnés sur `ontap-nas`, `ontap-nas-economy`, `ontap-nas-flexgroup` et `azure-netapp-files` backends.

Étape 1 : Configurer le StorageClass pour prendre en charge l'extension de volume

Pour redimensionner un volume persistant NFS, l'administrateur doit d'abord configurer la classe de stockage pour autoriser l'extension de volume en définissant le `allowVolumeExpansion` champ sur `true` :

```
cat storageclass-ontapnas.yaml
```

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontapnas
provisioner: csi.trident.netapp.io
parameters:
  backendType: ontap-nas
allowVolumeExpansion: true
```

Si vous avez déjà créé une classe de stockage sans cette option, vous pouvez simplement modifier la classe

de stockage existante en utilisant `kubectl edit storageclass` pour autoriser l'extension de volume.

Étape 2 : Créez un PVC avec le StorageClass que vous avez créé

```
cat pvc-ontapnas.yaml
```

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: ontapnas20mb
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 20Mi
  storageClassName: ontapnas
```

Trident devrait créer un PV NFS de 20 MiB pour ce PVC:

```
kubectl get pvc
NAME           STATUS    VOLUME
CAPACITY     ACCESS MODES   STORAGECLASS    AGE
ontapnas20mb   Bound    pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7   20Mi
RWO            ontapnas        9s

kubectl get pv pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7
NAME                                       CAPACITY   ACCESS MODES
RECLAIM POLICY   STATUS   CLAIM                  STORAGECLASS    REASON
AGE
pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7   20Mi       RWO
Delete           Bound    default/ontapnas20mb   ontapnas
2m42s
```

Étape 3 : Développer le PV

Pour redimensionner le PV nouvellement créé de 20 Mio à 1 Gio, modifiez le PVC et définissez `spec.resources.requests.storage` sur 1 Gio :

```
kubectl edit pvc ontapnas20mb
```

```
# Please edit the object below. Lines beginning with a '#' will be
ignored,
# and an empty file will abort the edit. If an error occurs while saving
this file will be
# reopened with the relevant failures.
#
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    pv.kubernetes.io/bind-completed: "yes"
    pv.kubernetes.io/bound-by-controller: "yes"
    volume.beta.kubernetes.io/storage-provisioner: csi.trident.netapp.io
  creationTimestamp: 2018-08-21T18:26:44Z
  finalizers:
  - kubernetes.io/pvc-protection
  name: ontapnas20mb
  namespace: default
  resourceVersion: "1958015"
  selfLink: /api/v1/namespaces/default/persistentvolumeclaims/ontapnas20mb
  uid: c1bd7fa5-a56f-11e8-b8d7-fa163e59eaab
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
# ...
```

Étape 4 : Valider l'expansion

Vous pouvez vérifier que le redimensionnement a fonctionné correctement en contrôlant la taille du PVC, du PV et du volume Trident :

```
kubectl get pvc ontapnas20mb
NAME           STATUS   VOLUME
CAPACITY   ACCESS MODES   STORAGECLASS    AGE
ontapnas20mb   Bound    pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7   1Gi
RWO            ontapnas        4m44s

kubectl get pv pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7
NAME                                       CAPACITY   ACCESS MODES
RECLAIM POLICY   STATUS   CLAIM                  STORAGECLASS    REASON
AGE
pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7   1Gi        RWO
Delete           Bound    default/ontapnas20mb   ontapnas
5m35s

tridentctl get volume pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7 -n trident
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
|                   NAME                  |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID              | STATE  | MANAGED |
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
| pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7 | 1.0 GiB | ontapnas      |
file     | c5a6f6a4-b052-423b-80d4-8fb491a14a22 | online | true    |
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
```

Comprendre les limites du sous-système RWX NVMe

ReadWriteMany (RWX) volumes utilisant le protocole NVMe ont une limite de scalabilité de 64 nœuds par volume. Ce qui suit inclut les limitations, explique l'architecture du sous-système NVMe concernée et décrit les étapes de résolution requises.

Comprendre la limite de 64 nœuds

Si vous prévoyez d'utiliser des volumes ReadWriteMany (RWX) avec le protocole NVMe, un seul volume NVMe RWX ne peut pas être monté par plus de 64 nœuds dans un cluster Kubernetes.

Ne planifiez pas de charges de travail qui montent le même RWX NVMe PersistentVolumeClaim sur plus de 64 nœuds.

Cette limitation s'applique uniquement aux volumes RWX qui utilisent le protocole NVMe.

Comprendre les modèles de sous-système NVMe

Modèle de sous-système par volume (Trident releases earlier than 26.02)

Dans les versions de Trident antérieures à 26.02, les volumes NVMe RWX sont provisionnés à l'aide d'un

modèle de sous-système par volume. Chaque volume NVMe RWX est associé à son propre sous-système NVMe dédié sur ONTAP.

Ce modèle est simple, mais il a une limite d'évolutivité inférieure. Dans les grands clusters Kubernetes, les limites des contrôleurs de sous-système sont rapidement atteintes car chaque volume RWX consomme un sous-système dédié.

Modèle de super-sous-système (introduit dans Trident 26.02)

À partir de Trident 26.02, les volumes RWX NVMe utilisent un modèle de super-sous-système partagé. Plusieurs volumes RWX NVMe partagent le même sous-système NVMe.

Chaque super-sous-système prend en charge jusqu'à 1024 espaces de noms (volumes). Ce modèle améliore considérablement l'évolutivité des charges de travail RWX et réduit la probabilité d'atteindre les limites du sous-système ONTAP.

Chaque volume RWX NVMe prend en charge jusqu'à 64 nœuds.

Identifier les symptômes d'erreur

Si vous créez ou attachez des volumes RWX NVMe à grande échelle, vous pourriez observer des erreurs similaires aux suivantes :

```
Maximum number of controllers reached. No more controllers can be created.
```

Cette erreur indique que la limite du contrôleur du sous-système ONTAP NVMe a été atteinte.

Résoudre les erreurs de limite du sous-système

Pour dépasser les limitations des sous-systèmes par volume et profiter du modèle de super-sous-système, mettez à niveau vers Trident 26.02 ou une version ultérieure.

Mettre à niveau Trident pour appliquer le modèle de super-sous-système

Pour appliquer le modèle de super-sous-système aux volumes RWX NVMe :

1. Mettez à niveau Trident vers la version 26.02 ou ultérieure.
2. Réduisez à zéro tous les pods qui utilisent des volumes RWX NVMe.
3. Vérifiez qu'aucune charge de travail n'utilise activement les volumes RWX NVMe.
4. Augmentez à nouveau le nombre de pods.

Cette séquence de redémarrage garantit que les volumes RWX NVMe sont attachés à l'aide du modèle super-sous-système.

- Cette limitation s'applique uniquement aux volumes RWX qui utilisent le protocole NVMe.
- La limite de 64 nœuds s'applique par volume RWX NVMe.
- Les autres modes d'accès et les autres protocoles ne sont pas affectés.

Évolutivité du contrôleur

Trident introduit la scalabilité des contrôleurs grâce à une meilleure gestion de la concurrence entre plusieurs pilotes de stockage. Les clients peuvent identifier quels pilotes Trident prennent en charge la scalabilité des contrôleurs à disponibilité générale et quels pilotes sont disponibles en avant-première technique dans Trident 26.02. Cela permet de prendre des décisions de déploiement éclairées et d'assurer une gestion appropriée des risques pour les environnements Kubernetes évolutifs.

Concepts clés et définitions

Évolutivité du contrôleur

La scalabilité du contrôleur fait référence à la capacité du contrôleur Trident à traiter plusieurs opérations de stockage en parallèle, au lieu de les sérialiser derrière un seul verrou. Ces opérations comprennent la création, la suppression et le redimensionnement de volumes, la création et la suppression d'instantanés, la publication et la dépublication de volumes, ainsi que la gestion du backend.

Lorsque la mise à l'échelle du contrôleur est activée, les opérations sur différents volumes et backends s'exécutent simultanément. Cela augmente le débit et réduit le temps d'exécution global dans les environnements comportant un grand nombre d'opérations PersistentVolumeClaim et VolumeSnapshot simultanées.

Prise en charge de l'évolutivité du contrôleur

Trident prend en charge l'évolutivité du contrôleur avec différents niveaux de maturité en fonction du pilote de stockage.

Disponibilité générale

Les pilotes suivants prennent en charge la mise à l'échelle du contrôleur à la disponibilité générale dans Trident 26.02 :

- `ontap-san`
- `ontap-nas`
- `google-cloud-netapp-volumes`

REMARQUE	Les `google-cloud-netapp-volumes` et `google-cloud-netapp-volumes-san` pilotes sont différents. Seul `google-cloud-netapp-volumes` est pris en charge. N'utilisez pas `google-cloud-netapp-volumes-san` dans les configurations ou exemples backend.

Activer la scalabilité du contrôleur

La scalabilité du contrôleur est contrôlée par l'option de configuration `enableConcurrency`. Cette option doit être explicitement activée lors de l'installation de Trident ou lors de la mise à jour d'un déploiement existant.

Déploiement de l'opérateur Trident

Pour activer la scalabilité du contrôleur avec l'opérateur Trident, définissez `enableConcurrency` sur `true` dans la ressource personnalisée `TridentOrchestrator`.

Nouvelle installation

Créez ou modifiez le `TridentOrchestrator` CR avec `enableConcurrency` défini sur `true` :

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  namespace: trident
  enableConcurrency: true
```

Appliquez le CR :

```
kubectl apply -f tridentorchestrator_cr.yaml
```

Installation existante

Corrigez le `TridentOrchestrator` CR existant pour activer l'évolutivité du contrôleur :

```
kubectl patch torc trident --type=merge -p
'{"spec":{"enableConcurrency":true}}'
```

Vérifiez que le paramètre a bien été appliqué :

```
kubectl get torc trident -o
jsonpath='{.status.currentInstallationParams.enableConcurrency}'
```

Déploiement Helm

Pour activer la scalabilité du contrôleur avec Helm, définissez la valeur `enableConcurrency` sur `true`.

Nouvelle installation

```
helm install trident netapp-trident/trident-operator --namespace trident
--create-namespace --set enableConcurrency=true
```

Installation existante

```
helm upgrade trident netapp-trident/trident-operator --namespace trident
--set enableConcurrency=true
```

Vous pouvez également définir `enableConcurrency` sur `true` dans un fichier `values.yaml` personnalisé :

```
# values.yaml
enableConcurrency: true
```

Installez ou mettez à niveau en utilisant le fichier de valeurs :

```
helm install trident netapp-trident/trident-operator --namespace trident
--create-namespace -f values.yaml
```

déploiement tridentctl

Pour activer la scalabilité du contrôleur avec `tridentctl`, transmettez le drapeau `--enable-concurrency` lors de l'installation.

Nouvelle installation

```
tridentctl install -n trident --enable-concurrency
```

Installation existante

Pour activer la mise à l'échelle du contrôleur sur un déploiement existant basé sur `tridentctl`, désinstallez et réinstallez avec l'indicateur :

```
tridentctl uninstall -n trident
tridentctl install -n trident --enable-concurrency
```

Vérifiez que la mise à l'échelle du contrôleur est activée

Après avoir activé la mise à l'échelle du contrôleur, vérifiez que le contrôleur Trident fonctionne avec la concurrence activée en consultant les journaux du pod du contrôleur :

```
kubectl logs -n trident deploy/trident-controller | grep -i concurrency
```

Vous devriez voir une entrée de journal indiquant que la concurrence est activée.

Aperçu technique

Les pilotes suivants prennent en charge la mise à l'échelle du contrôleur en tant qu'aperçu technique dans Trident 26.02 :

- `nas-eco`
- `san-eco`

Pour ces pilotes :

- La concurrence des contrôleurs est disponible pour l'évaluation et les tests
- Le comportement peut changer dans les prochaines versions
- L'utilisation en environnement de production n'est pas recommandée

Comportement de concurrence

Lorsque la mise à l'échelle du contrôleur est activée :

- Trident remplace le verrou global unique par un verrouillage fin, par ressource
- Les opérations qui modifient la même ressource sont sérialisées afin de garantir la cohérence des données
- Les opérations qui ne font que lire à partir d'une ressource peuvent s'exécuter simultanément avec d'autres opérations de lecture sur cette ressource
- Trident limite le nombre de requêtes API ONTAP simultanées à 20 par LIF de gestion afin d'éviter la surcharge des systèmes de stockage backend
- Si plusieurs backends partagent la même interface logique de gestion (LIF), ils partagent cette limite de 20 requêtes

Limitations et considérations connues

Les considérations suivantes s'appliquent à l'évolutivité du contrôleur :

- La concurrence est gérée en interne par le contrôleur Trident
- Il n'y a pas de limites de concurrence configurables par l'utilisateur dans cette version
- Le débit global dépend de :
  - Le pilote de stockage utilisé
  - Réactivité du backend
  - Performance du serveur API Kubernetes
- Une forte concurrence peut augmenter la charge sur les systèmes de stockage backend

Avertissements et limitations

Les limitations suivantes s'appliquent dans Trident 26.02 :

- Le comportement de mise à l'échelle du contrôleur n'est pas identique pour tous les pilotes
- Les pilotes en préversion technique peuvent présenter :
  - Performances irrégulières sous forte charge
  - Changements de comportement entre les versions
- Le débogage des opérations simultanées peut être plus complexe en raison de l'exécution parallèle
- Les indicateurs et les journaux peuvent afficher des résultats d'opérations entrelacées

Recommandations

- Utilisez les pilotes à disponibilité générale (GA) pour les environnements de production nécessitant une évolutivité élevée

- Évaluer les pilotes en préversion technique dans des environnements hors production
- Surveillez les performances du backend et du contrôleur lors du fonctionnement à grande échelle
- Évitez de présumer l'ordre des opérations dans les scripts d'automatisation

Extension automatique du volume

L'extension automatique des volumes permet aux Persistent Volumes provisionnés par Trident d'augmenter automatiquement leur capacité lorsque celle-ci atteint un seuil défini. Cette fonctionnalité réduit les coûts d'exploitation et contribue à prévenir les interruptions d'application dues à la saturation de la capacité. L'extension automatique du volume est mise en œuvre à l'aide des Autogrow Policies. Une Autogrow Policy définit :

- Le seuil d'utilisation qui déclenche l'expansion
- La quantité dont le volume augmente
- La taille maximale que le volume peut atteindre

REMARQUE	Les volumes augmentent automatiquement lorsque le seuil d'utilisation défini est dépassé. Les volumes ne sont jamais réduits automatiquement.

Exigences

Avant de configurer l'extension automatique du volume, assurez-vous que les exigences suivantes sont remplies :

- Trident 26.02 ou version ultérieure
- Autorisations de contrôle d'accès basées sur les rôles pour créer `TridentAutogrowPolicy` des ressources personnalisées
- StorageClasses configuré avec `allowVolumeExpansion: true`
- Protocoles ONTAP pris en charge :
  - Système de fichiers réseau (NFS)
  - Internet Small Computer Systems Interface (iSCSI)
  - Non-Volatile Memory Express (NVMe)
  - Protocole Fibre Channel (FCP)

Limitations

- Les volumes de blocs bruts ONTAP Non-Volatile Memory Express antérieurs à ONTAP 9.16.1 ne prennent pas en charge l'extension automatique.
- Pour les volumes de réseau de stockage, si `growthAmount` est inférieur ou égal à 50 mébioctets, Trident augmente automatiquement la valeur à 51 mébioctets avant le redimensionnement, à condition que la taille résultante ne dépasse pas `maxSize`.
- Dans les environnements brownfield, l'extension automatique peut ne pas fonctionner pour certains volumes existants en raison du comportement de migration de la publication des volumes.
- Lorsqu'un volume atteint `maxSize`, aucune expansion supplémentaire ne se produit.
- Protocoles pris en charge pour l'extension automatique du volume :

- Système de fichiers réseau (NFS)
- Internet Small Computer Systems Interface (iSCSI)
- Non-Volatile Memory Express (NVMe)
- Protocole Fibre Channel (FCP)

Provisionnement de volumes avec stratégie d'extension automatique

La politique Autogrow peut être configurée à deux niveaux :

- Niveau de classe de stockage : définit la valeur par défaut pour tous les volumes (à l'aide d'une annotation)
- Niveau PVC : Remplace la valeur par défaut de la classe de stockage (en utilisant une annotation)

Créer une politique Autogrow

Les politiques Autogrow permettent l'expansion automatique des volumes lorsque ceux-ci atteignent un seuil de capacité défini.

Assurez-vous d'avoir :

- Trident 26.02 ou version ultérieure installé
- Autorisations de contrôle d'accès basées sur les rôles pour créer `TridentAutogrowPolicy` des ressources
- Compréhension des exigences de croissance de la charge de travail

Une politique Autogrow définit comment les volumes s'étendent automatiquement lorsqu'ils atteignent un seuil de capacité défini.

Vous pouvez créer des stratégies Autogrow à tout moment de votre flux de travail :

- Avant que les StorageClasses et les volumes ne soient créés
- Après que les StorageClasses existent
- Une fois les volumes provisionnés

Cette flexibilité vous permet d'introduire une extension automatique sans recréer les ressources existantes.

Spécifications de la politique Autogrow

Les politiques d'autogrow sont des ressources personnalisées Kubernetes définies comme suit :

Champ	Description	Format	Obligatoire	Exemple	Défaut
nom	Identifiant unique de la politique	Chaîne	Oui	production-db-policy	Aucune
usedThreshold	Pourcentage de capacité qui déclenche l'expansion	Chaîne de pourcentage	Oui	"80%"	Aucune
growthAmount	Montant à augmenter lorsque le seuil est atteint	Pourcentage ou taille	Non	« 10% » ou « 5Gi »	"10%"

Champ	Description	Format	Obligatoire	Exemple	Défaut
maxSize	Limite de taille maximale du volume	quantité Kubernetes	Non	"500Gi"	Illimité

Créer une politique Autogrow

Étapes

1. Créez un fichier YAML qui définit votre Autogrow Policy :

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: standard-autogrow
spec:
  usedThreshold: "80%"
  growthAmount: "10%"
  maxSize: "500Gi"
```

2. Appliquez la politique à votre cluster :

```
kubectl apply -f autogrow-policy.yaml
```

3. Vérifiez que la policy a été créée :

```
kubectl get tridentautogrowpolicy standard-autogrow
```

Résultat attendu

```
NAME                USED THRESHOLD   GROWTH AMOUNT   STATE
standard-autogrow   80%              10%             Success
```

États de la stratégie

Après avoir créé une politique, Trident valide la spécification et lui attribue l'un des états suivants :

État	Description	Action requise
Succès	La stratégie est validée et prête à l'emploi.	Aucun.
Échec	Des erreurs de validation ont été détectées.	Examinez et corrigez la spécification.
Suppression	Suppression en cours.	Attendez la fin.

Associer une politique à une StorageClass

Vous pouvez associer une stratégie d'autogrow à un StorageClass en utilisant l'annotation `trident.netapp.io/autogrowPolicy`. Tous les volumes provisionnés à partir de ce StorageClass héritent de la stratégie.

REMARQUE	Le StorageClass doit avoir `allowVolumeExpansion: true`.

Étapes

1. Créez ou modifiez un StorageClass avec l'annotation de stratégie d'extension automatique :

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-gold
  annotations:
    trident.netapp.io/autogrowPolicy: "production-db-policy"
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
  fsType: "ext4"
allowVolumeExpansion: true
```

2. Appliquez le StorageClass :

```
kubectl apply -f storageclass.yaml
```

3. Vérifiez l'annotation :

```
kubectl get storageclass ontap-gold -o
jsonpath='{.metadata.annotations.trident\.netapp\.io/autogrowPolicy}'
```

Résultat attendu

```
production-db-policy
```

Précédence de la politique

Lorsque des annotations de stratégie d'extension automatique sont définies à la fois sur un StorageClass et un PVC, Trident applique les règles de priorité suivantes :

1. **L'annotation PVC est prioritaire.** Si un PVC définit `trident.netapp.io/autogrowPolicy`, cette valeur est toujours utilisée.
2. **StorageClass annotation s'applique uniquement lorsque le PVC ne comporte aucune annotation.**

3. **Si aucune des deux ne comporte l'annotation,** aucune Autogrow Policy n'est appliquée.

Annotation StorageClass	Annotation PVC	Comportement efficace
`trident.netapp.io/autogrowPolicy: standard-agp`	Non défini	Utilisations `standard-agp`.
`trident.netapp.io/autogrowPolicy: standard-agp`	`trident.netapp.io/autogrowPolicy: logs-policy`	Utilise `logs-policy` (les PVC remplacent StorageClass).
`trident.netapp.io/autogrowPolicy: standard-agp`	`trident.netapp.io/autogrowPolicy: "none"`	Aucune politique d'autogrow (PVC désactive l'autogrow).
Non défini	`trident.netapp.io/autogrowPolicy: dev-policy`	Utilisations `dev-policy`.
Non défini	Non défini	Aucune politique d'autocroissance.

Exemples de configuration

Les exemples suivants montrent les configurations courantes d'Autogrow Policy pour différents cas d'usage.

Politique conservatrice pour les bases de données de production

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: production-db-policy
spec:
  usedThreshold: "75%"
  growthAmount: "20%"
  maxSize: "5Ti"
```

Stockage de journaux avec incréments de croissance fixes

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: log-storage-policy
spec:
  usedThreshold: "90%"
  growthAmount: "10Gi"
  maxSize: "100Gi"
```

Politique minimale avec valeurs par défaut

Lorsque vous omettez `growthAmount` et `maxSize`, Trident utilise les valeurs par défaut (`10%` (croissance, taille illimitée) :

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: logs-policy
spec:
  usedThreshold: "85%"
```

Stratégie avec une valeur personnalisée maxSize et la valeur par défaut growthAmount

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: default-ga-policy
spec:
  usedThreshold: "70%"
  maxSize: "100Gi"
```

Croissance agressive avec maxSize illimité

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: aggressive-growth-policy
spec:
  usedThreshold: "80%"
  growthAmount: "150%"
```

Politique avec des pourcentages fractionnaires

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: precise-policy
spec:
  usedThreshold: "80.28%"
  growthAmount: "10.65%"
  maxSize: "100Gi"
```

REMARQUE	Les pourcentages fractionnaires sont pris en charge. Si vous spécifiez plus de trois décimales, Trident arrondit la valeur à trois décimales.

NAS StorageClass avec Autogrow

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-nas-autogrow
  annotations:
    trident.netapp.io/autogrowPolicy: "standard-autogrow"
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-nas"
  fsType: "ext4"
allowVolumeExpansion: true
```

SAN StorageClass avec Autogrow

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: database-storage
  annotations:
    trident.netapp.io/autogrowPolicy: "production-db-policy"
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
  fsType: "ext4"
allowVolumeExpansion: true
```

Gérer les stratégies Autogrow

Après avoir créé des stratégies Autogrow, vous pouvez les consulter, les mettre à jour et les supprimer selon vos besoins. Vous pouvez également surveiller quels volumes utilisent une stratégie donnée.

Afficher les politiques Autogrow

Lister toutes les politiques

Utilisez `kubectl` pour lister toutes les stratégies d'extension automatique de votre cluster :

```
kubectl get tridentautogrowpolicy
```

Vous pouvez également utiliser `tridentctl` :

```
tridentctl get autogrowpolicy
```

Afficher les détails de la politique

Pour consulter la spécification complète et l'état d'une politique :

```
kubectl describe tridentautogrowpolicy production-db-policy
```

Pour afficher une règle avec ses volumes associés au format YAML :

```
tridentctl get autogrowpolicy production-db-policy -o yaml
```

Mettre à jour une politique Autogrow

Vous pouvez modifier une politique existante pour changer son seuil, son taux de croissance ou sa taille maximale. Les modifications prennent effet immédiatement pour tous les volumes qui utilisent la politique.

IMPORTANT	Les modifications affectent tous les volumes utilisant actuellement cette politique. Testez d'abord les modifications dans un environnement hors production lorsque cela est possible.

Étapes

1. Modifier la politique :

   ```
   kubectl edit tridentautogrowpolicy production-db-policy
   ```

2. Modifiez les `spec` champs selon vos besoins :

   ```
   spec:
     usedThreshold: "75%"     # Changed from 80%
     growthAmount: "20%"      # Changed from 10%
     maxSize: "1Ti"           # Changed from 500Gi
   ```

3. Enregistrez et quittez. Les modifications prennent effet immédiatement.

Considérations relatives à la mise à jour

- **Effet immédiat :** Tous les volumes utilisant la politique adoptent de nouveaux paramètres lors de la prochaine évaluation de la croissance.
- **Aucun redémarrage du volume nécessaire :** Les modifications s'appliquent à la prochaine opération de croissance.

- **Testez d'abord :** Validez les modifications dans un environnement non production lorsque cela est possible.
- **Communiquer les changements :** Informez les équipes lorsque vous modifiez des politiques partagées.

Supprimer une politique Autogrow

Les politiques d'autogrow utilisent la protection du finaliseur pour empêcher toute suppression accidentelle pendant que des volumes les utilisent activement.

Étapes

1. Supprimez la règle :

```
kubectl delete tridentautogrowpolicy production-db-policy
```

2. Si des volumes utilisent encore cette stratégie, la suppression entre dans un état `Deleting`. Vérifiez quels volumes sont concernés :

```
tridentctl get autogrowpolicy production-db-policy -o yaml
```

3. Supprimez la stratégie de chaque volume concerné. Choisissez l'une des options suivantes :

   - **Option A : Désactiver explicitement la croissance automatique** en définissant l'annotation sur `"none"`:

```
kubectl annotate pvc <pvc-name> \
  trident.netapp.io/autogrowPolicy="none" \
  --overwrite
```

   - **Option B: Supprimer complètement l'annotation :**

```
kubectl annotate pvc <pvc-name> \
  trident.netapp.io/autogrowPolicy-
```

Comportement de suppression

Scénario	Comportement
Aucun volume n'utilise la politique	La règle est supprimée immédiatement.
Les volumes utilisent la politique	La stratégie entre dans `Deleting` état. Un finaliseur bloque la finalisation jusqu'à ce que tous les volumes soient supprimés.
Tous les volumes sont supprimés de la stratégie	Les finalizers sont supprimés et la policy est supprimée.

Surveiller l'utilisation de la politique Autogrow

Vérifiez les volumes à l'aide d'une règle

```
tridentctl get autogrowpolicy production-db-policy -o json | jq '.volumes'
```

Découvrez quelle politique un volume utilise

```
kubectl get pvc database-pvc -o
jsonpath='{.metadata.annotations.trident\.netapp\.io/autogrowPolicy}'
```

Surveiller les événements de stratégie

```
kubectl get events --field-selector
involvedObject.kind=TridentAutogrowPolicy
```

Protocoles pris en charge

Autogrow prend en charge les protocoles de stockage suivants :

- NFS
- iSCSI
- FCP
- NVMe

REMARQUE	Pour les volumes SAN, si la valeur configurée `growthAmount` est de 50 Mio ou moins, Trident augmente automatiquement la marge de croissance à 51 Mo pour l'opération de redimensionnement, tant que la taille résultante ne dépasse pas `maxSize`.

Limitations connues

- **Volumes de blocs bruts NVMe ONTAP :** Les volumes créés avec des versions d'ONTAP antérieures à 9.16.1 ne prennent pas en charge l'autogrow.
- **Volumes existants (déploiements brownfield) :** L'extension automatique peut ne pas fonctionner pour les volumes existants, même si une stratégie d'extension automatique valide est appliquée. Ceci est dû à une migration en cours des publications de volumes. Pour confirmer que la migration est terminée, vérifiez les journaux du contrôleur Trident pour les messages `"Migration completed"`.

Foire aux questions

À quel moment Trident évalue le seuil ?

Trident surveille en permanence l'utilisation du volume. Lorsque la capacité utilisée dépasse le `usedThreshold`, Trident crée une demande de redimensionnement interne et augmente le volume de la valeur configurée `growthAmount`.

Par exemple, cette politique déclenche une extension à 80% de la capacité et augmente le volume de 10% à chaque fois, jusqu'à un maximum de 500 GiB :

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: standard-autogrow
spec:
  usedThreshold: "80%"
  growthAmount: "10%"
  maxSize: "500Gi"
```

Puis-je appliquer une stratégie après que les volumes ont déjà été provisionnés ?

Oui. Vous pouvez créer une stratégie d'autogrow à tout moment et l'appliquer aux PVC existants en ajoutant ou en mettant à jour l' `trident.netapp.io/autogrowPolicy`annotation. Vous n'avez pas besoin de recréer le PVC ni le StorageClass.

Pour appliquer une politique à un PVC existant :

```
kubectl annotate pvc <pvc-name> \
  trident.netapp.io/autogrowPolicy="production-db-policy" \
  --overwrite
```

Pour appliquer une politique à un StorageClass existant :

```
kubectl annotate storageclass ontap-gold \
  trident.netapp.io/autogrowPolicy="production-db-policy" \
  --overwrite
```

Que se passe-t-il si je configure une stratégie d'autogrow à la fois sur le StorageClass et sur le PVC ?

L'annotation PVC est toujours prioritaire. Si un PVC possède l' `trident.netapp.io/autogrowPolicy`annotation, Trident utilise cette valeur, quelle que soit la valeur spécifiée par la StorageClass. Consultez "Précédence de la politique" pour plus de détails.

Par exemple, étant donné ce StorageClass :

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-gold
  annotations:
    trident.netapp.io/autogrowPolicy: "standard-agp"
provisioner: csi.trident.netapp.io
allowVolumeExpansion: true
```

Et ce PVC qui annule la politique StorageClass :

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: database-pvc
  annotations:
    trident.netapp.io/autogrowPolicy: "logs-policy"
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 50Gi
  storageClassName: ontap-gold
```

Trident utilise `logs-policy` pour `database-pvc`, pas `standard-agp`.

Comment désactiver l'autogrow pour un volume spécifique ?

Définissez l'annotation PVC sur `"none"`. Cela remplace toute politique au niveau de la StorageClass pour ce volume :

```
kubectl annotate pvc <pvc-name> \
  trident.netapp.io/autogrowPolicy="none" \
  --overwrite
```

Vous pouvez vérifier que la croissance automatique est désactivée :

```
kubectl get pvc <pvc-name> -o jsonpath
='{.metadata.annotations.trident\.netapp\.io/autogrowPolicy}'
```

Résultat attendu

```
none
```

Que se passe-t-il lorsqu'un volume atteint maxSize?

Trident cesse d'étendre le volume. Aucune autre demande de redimensionnement n'est créée pour ce volume, même si l'utilisation continue d'augmenter au-delà de la `usedThreshold`.

Par exemple, avec cette politique, Trident cesse d'augmenter le volume une fois qu'il atteint 100 GiB :

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: capped-policy
spec:
  usedThreshold: "90%"
  growthAmount: "10Gi"
  maxSize: "100Gi"
```

Pour autoriser une croissance illimitée, omettez `maxSize` ou définissez-la sur `0` :

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: unlimited-policy
spec:
  usedThreshold: "85%"
  growthAmount: "10%"
```

Puis-je modifier une stratégie sans redémarrer les volumes ?

Oui. Lorsque vous mettez à jour une stratégie, tous les volumes utilisant cette stratégie adoptent les nouveaux paramètres lors de la prochaine évaluation de la croissance. Aucun redémarrage de volume n'est requis.

Pour mettre à jour une politique en place :

```
kubectl edit tridentautogrowpolicy production-db-policy
```

Modifiez les champs si nécessaire :

```
spec:
  usedThreshold: "75%"     # Changed from 80%
  growthAmount: "20%"      # Changed from 10%
  maxSize: "1Ti"           # Changed from 500Gi
```

Enregistrez et quittez. Vérifiez la politique mise à jour :

```
kubectl get tridentautogrowpolicy production-db-policy
```

Résultat attendu

```
NAME                   USED THRESHOLD   GROWTH AMOUNT   STATE
production-db-policy   75%              20%             Success
```

Pourquoi ma stratégie est-elle en état d'échec ?

Un `Failed` état indique que la spécification de la politique contient des erreurs de validation. Exécutez la commande suivante pour afficher les détails de l'erreur :

```
kubectl describe tridentautogrowpolicy <policy-name>
```

Les causes fréquentes incluent une valeur invalide `usedThreshold` (doit être comprise entre 1 et 99 %), une `growthAmount` qui dépasse `maxSize`, ou un format de quantité Kubernetes invalide. Corrigez la spécification et réappliquez :

```
kubectl apply -f autogrow-policy.yaml
```

Pourquoi ne puis-je pas supprimer une policy ?

Les politiques utilisent une protection de finalisation. Si des volumes utilisent encore la politique, la suppression entre dans un `Deleting` état et attend que tous les volumes soient supprimés de la politique.

Identifiez les volumes concernés :

```
tridentctl get autogrowpolicy production-db-policy -o yaml
```

Supprimez ensuite l'annotation de chaque PVC :

```
# Option A: Explicitly disable autogrow
kubectl annotate pvc <pvc-name> \
  trident.netapp.io/autogrowPolicy="none" \
  --overwrite

# Option B: Remove the annotation entirely
kubectl annotate pvc <pvc-name> \
  trident.netapp.io/autogrowPolicy-
```

Après la suppression de tous les volumes, le finaliseur est libéré et la policy est supprimée.

La fonction d'extension automatique fonctionne-t-elle avec tous les backends ONTAP ?

La fonctionnalité d'extension automatique prend en charge les protocoles NFS, iSCSI, FCP et NVMe. Cependant, les volumes de blocs bruts NVMe nécessitent ONTAP 9.16.1 ou une version ultérieure.

Les volumes existants dans les déploiements brownfield peuvent nécessiter la migration de la publication des volumes avant que l'autogrow ne prenne effet. Vérifiez l'état de la migration en consultant les journaux du contrôleur Trident :

```
kubectl logs -l app=trident-controller -n trident | grep "Migration
completed"
```

Les exemples suivants de StorageClass montrent l'extension automatique configurée pour les backends NAS et SAN :

Backend NAS

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-nas-autogrow
  annotations:
    trident.netapp.io/autogrowPolicy: "standard-autogrow"
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-nas"
  fsType: "ext4"
allowVolumeExpansion: true
```

Backend SAN

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: database-storage
  annotations:
    trident.netapp.io/autogrowPolicy: "production-db-policy"
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
  fsType: "ext4"
allowVolumeExpansion: true
```

Quel est le montant minimal de croissance pour les volumes SAN ?

Pour les volumes SAN, la marge de croissance minimale effective est de 51 Mo. Si vous configurez une `growthAmount` de 50 Mio ou moins, Trident augmente automatiquement la croissance à 51 Mo pour l'opération de redimensionnement.

Par exemple, cette politique définit une `growthAmount` de `"40Mi"`, mais Trident applique une croissance de 51 Mo à tout volume SAN qui l'utilise :

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: san-minimal-policy
spec:
  usedThreshold: "85%"
  growthAmount: "40Mi"
  maxSize: "100Gi"
```

Pour éviter ce réglage automatique, définissez `growthAmount` sur une valeur supérieure à 50 Mio :

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: san-policy
spec:
  usedThreshold: "85%"
  growthAmount: "100Mi"
  maxSize: "500Gi"
```

Importer des volumes

Vous pouvez importer des volumes de stockage existants en tant que PV Kubernetes en utilisant `tridentctl import` ou en créant une revendication de volume persistant (PVC) avec des annotations d'importation Trident.

Aperçu et considérations

Vous pouvez importer un volume dans Trident pour :

- Conteneurisez une application et réutilisez son ensemble de données existant
- Utilisez un clone d'un ensemble de données pour une application éphémère
- Reconstruire un cluster Kubernetes défaillant
- Migrer les données d'application lors d'une reprise après sinistre

Considérations

Avant d'importer un volume, examinez les considérations suivantes.

- Trident peut importer uniquement les volumes ONTAP de type RW (lecture-écriture). Les volumes de type DP (protection des données) sont des volumes de destination SnapMirror. Vous devez rompre la relation de miroir avant d'importer le volume dans Trident.
- Nous suggérons d'importer les volumes sans connexion active. Pour importer un volume utilisé activement, clonez le volume, puis effectuez l'importation.

AVERTISSEMENT	Ceci est particulièrement important pour les volumes de blocs, car Kubernetes ignorerait la connexion précédente et pourrait facilement associer un volume actif à un pod. Cela peut entraîner une corruption des données.

- Bien que `StorageClass` doive être spécifié sur un PVC, Trident ne l'utilise pas lors de l'importation. Les classes de stockage sont utilisées lors de la création du volume pour sélectionner parmi les pools disponibles en fonction des caractéristiques de stockage. Parce que le volume existe déjà, aucune sélection de pool n'est requise lors de l'importation. Par conséquent, l'importation ne sera pas un échec même si le volume existe sur un backend ou un pool qui ne correspond pas à la classe de stockage spécifiée dans le PVC.
- La taille du volume existant est déterminée et définie dans le PVC. Après que le volume a été importé par le pilote de stockage, le PV est créé avec un ClaimRef vers le PVC.
  - La politique de récupération est initialement définie sur `retain` dans le PV. Après que Kubernetes a correctement lié le PVC et le PV, la politique de récupération est mise à jour pour correspondre à la politique de récupération de la Storage Class.
  - Si la politique de récupération de la Storage Class est `delete`, le volume de stockage sera supprimé lorsque le PV sera supprimé.
- Par défaut, Trident gère le PVC et renomme le FlexVol volume et le LUN sur le backend. Vous pouvez passer l' `--no-manage`option pour importer un volume non géré et l' `--no-rename`option pour conserver le nom du volume.
  - `--no-manage`* - Si vous utilisez le `--no-manage` flag, Trident n'effectue aucune opération supplémentaire sur le PVC ou le PV pendant tout le cycle de vie des objets. Le volume de stockage n'est pas supprimé lorsque le PV est supprimé et d'autres opérations telles que le clonage de volume et le redimensionnement de volume sont également ignorées.

◦ `--no-rename`* - Si vous utilisez le `--no-rename` indicateur, Trident conserve le nom du volume existant lors de l'importation des volumes et gère le cycle de vie des volumes. Cette option est prise en charge uniquement pour les `ontap-nas`, `ontap-san` (y compris les systèmes ASA r2), et les pilotes `ontap-san-economy`.

ASTUCE	Ces options sont utiles si vous souhaitez utiliser Kubernetes pour les charges de travail conteneurisées mais que vous souhaitez par ailleurs gérer le cycle de vie du volume de stockage en dehors de Kubernetes.

- Une annotation est ajoutée aux PVC et PV et sert à la fois à indiquer que le volume a été importé et si les PVC et PV sont gérés. Cette annotation ne doit pas être modifiée ni supprimée.

Importer un volume

Vous pouvez importer un volume en utilisant soit `tridentctl import` soit en créant un PVC avec des annotations d'importation Trident.

REMARQUE	Si vous utilisez des annotations PVC, vous n'avez pas besoin de télécharger ou d'utiliser `tridentctl` pour importer le volume.

Utilisation de tridentctl

Étapes

1. Créez un fichier PVC (par exemple, `pvc.yaml`) qui sera utilisé pour créer le PVC. Le fichier PVC doit inclure `name`, `namespace`, `accessModes` et `storageClassName`. Vous pouvez éventuellement spécifier `unixPermissions` dans la définition de votre PVC.

   Voici un exemple de spécification minimale :

   ```
   kind: PersistentVolumeClaim
   apiVersion: v1
   metadata:
     name: my_claim
     namespace: my_namespace
   spec:
     accessModes:
       - ReadWriteOnce
     storageClassName: my_storage_class
   ```

   REMARQUE N'indiquez que les paramètres requis. Les paramètres supplémentaires tels que le nom du PV ou la taille du volume peuvent entraîner l'échec de la commande d'importation.

2. Utilisez la commande `tridentctl import` pour spécifier le nom du backend Trident contenant le volume et le nom qui identifie de manière unique le volume sur le stockage (par exemple : ONTAP FlexVol, Element Volume). L'argument `-f` est obligatoire pour spécifier le chemin d'accès au fichier PVC.

   ```
   tridentctl import volume <backendName> <volumeName> -f <path-to-pvc-file>
   ```

Utilisation des annotations PVC

Étapes

1. Créez un fichier YAML PVC (par exemple, `pvc.yaml`) avec les annotations d'importation Trident requises. Le fichier PVC doit inclure :

   - `name` et `namespace` dans les métadonnées
   - `accessModes`, `resources.requests.storage`, et `storageClassName` dans les spécifications
   - Annotations :
     - `trident.netapp.io/importOriginalName`: Nom du volume sur le backend
     - `trident.netapp.io/importBackendUUID`: UUID du backend où le volume existe
     - `trident.netapp.io/notManaged` (*Facultatif*) : Définir sur `"true"` pour les volumes non gérés. La valeur par défaut est `"false"`.

Voici un exemple de spécification pour l'importation d'un volume géré :

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: <pvc-name>
  namespace: <namespace>
  annotations:
    trident.netapp.io/importOriginalName: "<volume-name>"
    trident.netapp.io/importBackendUUID: "<backend-uuid>"
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: <size>
  storageClassName: <storage-class-name>
```

2. Appliquez le fichier YAML PVC à votre cluster Kubernetes :

```
kubectl apply -f <pvc-file>.yaml
```

Trident importera automatiquement le volume et le liera au PVC.

Exemples

Consultez les exemples d'importation de volumes suivants pour les pilotes pris en charge.

ONTAP NAS et ONTAP NAS FlexGroup

Trident prend en charge l'importation de volumes à l'aide des `ontap-nas` et `ontap-nas-flexgroup` pilotes.

REMARQUE

- Trident ne prend pas en charge l'importation de volumes à l'aide du `ontap-nas-economy`driver.
- Les `ontap-nas` et `ontap-nas-flexgroup` pilotes n'autorisent pas les noms de volume en double.

Chaque volume créé avec le `ontap-nas` driver est un volume FlexVol sur le cluster ONTAP. L'importation de volumes FlexVol avec le `ontap-nas` driver fonctionne de la même manière. Un volume FlexVol qui existe déjà sur un cluster ONTAP peut être importé comme un `ontap-nas` PVC. De même, les volumes FlexGroup peuvent être importés comme des `ontap-nas-flexgroup` PVC.

Exemples ONTAP NAS utilisant tridentctl

Les exemples suivants montrent comment importer des volumes gérés et non gérés à l'aide de `tridentctl`.

Volume géré

L'exemple suivant importe un volume nommé `managed_volume` sur un backend nommé `ontap_nas` :

```
tridentctl import volume ontap_nas managed_volume -f <path-to-pvc-file>

+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+---------+---------+
|                   NAME                  |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID              | STATE  | MANAGED |
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+---------+---------+
| pvc-bf5ad463-afbb-11e9-8d9f-5254004dfdb7 | 1.0 GiB | standard      |
file     | c5a6f6a4-b052-423b-80d4-8fb491a14a22 | online | true    |
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+---------+---------+
```

Volume non géré

Lors de l'utilisation de l' `--no-manage`argument, Trident ne renomme pas le volume.

L'exemple suivant importe `unmanaged_volume` sur le `ontap_nas` backend :

```
tridentctl import volume nas_blog unmanaged_volume -f <path-to-pvc-
file> --no-manage

+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+---------+---------+
|                   NAME                  |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID              | STATE  | MANAGED |
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+---------+---------+
| pvc-df07d542-afbc-11e9-8d9f-5254004dfdb7 | 1.0 GiB | standard      |
file     | c5a6f6a4-b052-423b-80d4-8fb491a14a22 | online | false   |
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+---------+---------+
```

Exemples ONTAP NAS utilisant des annotations PVC

Les exemples suivants montrent comment importer des volumes gérés et non gérés à l'aide d'annotations PVC.

Volume géré

L'exemple suivant importe un volume de 1 GiB `ontap-nas` nommé `ontap_volume1` à partir du backend `81abcb27-ea63-49bb-b606-0a5315ac5f21` avec le mode d'accès RWO défini à l'aide d'annotations PVC :

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: <managed-imported-volume>
  namespace: <namespace>
  annotations:
    trident.netapp.io/importOriginalName: "ontap_volume1"
    trident.netapp.io/importBackendUUID: "81abcb27-ea63-49bb-b606-
0a5315ac5f21"
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: <storage-class-name>
```

Volume non géré

L'exemple suivant importe 1Gi `ontap-nas` volume nommé `ontap-volume2` depuis le backend `34abcb27-ea63-49bb-b606-0a5315ac5f34` avec le mode d'accès RWO défini à l'aide d'annotations PVC :

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: <umanaged-imported-volume>
  namespace: <namespace>
  annotations:
    trident.netapp.io/importOriginalName: "ontap-volume2"
    trident.netapp.io/importBackendUUID: "34abcb27-ea63-49bb-b606-
0a5315ac5f34"
    trident.netapp.io/notManaged: "true"
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: <storage-class-name>
```

ONTAP SAN

Trident prend en charge l'importation de volumes à l'aide des `ontap-san` (iSCSI, NVMe/TCP et FC) et `ontap-san-economy` pilotes.

Trident peut importer des volumes SAN ONTAP FlexVol qui contiennent un seul LUN. Cela est cohérent avec le `ontap-san` driver, qui crée un volume FlexVol pour chaque PVC et un LUN dans le volume FlexVol. Trident importe le volume FlexVol et l'associe à la définition du PVC. Trident peut importer des volumes `ontap-san-economy` qui contiennent plusieurs LUN.

Les exemples suivants montrent comment importer des volumes gérés et non gérés :

Volume géré

Pour les volumes gérés, Trident renomme le volume FlexVol au format `pvc-<uuid>` et le LUN à l'intérieur du volume FlexVol en `lun0`.

L'exemple suivant importe le `ontap-san-managed` volume FlexVol qui est présent sur le `ontap_san_default` backend :

```
tridentctl import volume ontapsan_san_default ontap-san-managed -f pvc-
basic-import.yaml -n trident -d

+-----------------------------------------+--------+--------------
+----------+-----------------------------------------+--------+---------+
|                   NAME                  |  SIZE  | STORAGE CLASS |
PROTOCOL |             BACKEND UUID              | STATE  | MANAGED |
+-----------------------------------------+--------+--------------
+----------+-----------------------------------------+--------+---------+
| pvc-d6ee4f54-4e40-4454-92fd-d00fc228d74a | 20 MiB | basic         |
block    | cd394786-ddd5-4470-adc3-10c5ce4ca757 | online | true    |
+-----------------------------------------+--------+--------------
+----------+-----------------------------------------+--------+---------+
```

Volume non géré

L'exemple suivant importe `unmanaged_example_volume` sur le `ontap_san` backend :

```
tridentctl import volume -n trident san_blog unmanaged_example_volume
-f pvc-import.yaml --no-manage
+-----------------------------------------+---------+--------------
+----------+-----------------------------------------+--------+---------+
|                   NAME                  |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID              | STATE  | MANAGED |
+-----------------------------------------+---------+--------------
+----------+-----------------------------------------+--------+---------+
| pvc-1fc999c9-ce8c-459c-82e4-ed4380a4b228 | 1.0 GiB | san-blog      |
block    | e3275890-7d80-4af6-90cc-c7a0759f555a | online | false   |
+-----------------------------------------+---------+--------------
+----------+-----------------------------------------+--------+---------+
```

Si vous avez des LUN mappés à des igroups qui partagent un IQN avec l'IQN d'un nœud Kubernetes, comme illustré dans l'exemple suivant, vous recevrez l'erreur : `LUN already mapped to initiator(s) in this group`. Vous devrez supprimer l'initiateur ou dissocier le LUN pour importer le volume.

```
Vserver   Igroup       Protocol OS Type  Initiators
--------- ------------ -------- -------- ------------------------------------
svm0      k8s-nodename.example.com-fe5d36f2-cded-4f38-9eb0-c7719fc2f9f3
                       iscsi    linux    iqn.1994-05.com.redhat:4c2e1cf35e0
svm0      unmanaged-example-igroup
                       mixed    linux    iqn.1994-05.com.redhat:4c2e1cf35e0
```

Exemples d'économie SAN ONTAP

Les exemples suivants montrent comment importer des volumes gérés et non gérés pour le backend `ontap-san-economy`.

Volume géré

Lors de l'importation d'un volume géré, Trident prend possession du FlexVol et le renomme. Vous devez tenir compte de ce renommage lors de l'importation de plusieurs LUN provenant du même FlexVol.

L'exemple suivant importe `lun1` du FlexVol `toimport` en tant que volume géré nommé `vol-managed-saneco` :

```
tridentctl import volume vol-managed-saneco toimport/lun1 -f
import1.yaml
```

Après l'importation `lun1`, Trident renomme le FlexVol (par exemple, en `trident_lun_pool_xyz`). Pour importer d'autres LUNs du même FlexVol, utilisez le nouveau nom FlexVol :

```
tridentctl import volume vol-managed-saneco trident_lun_pool_xyz/lun2
-f import2.yaml
```

REMARQUE	Le `ontap-san-economy` backend importe une LUN à la fois. Vous pouvez automatiser plusieurs importations à l'aide d'un script.

Volume non géré

Lorsque vous importez un volume non géré, Trident ne prend pas possession du FlexVol®. Cependant, le FlexVol® et la LUN doivent suivre les conventions de nommage Trident.

Format de nommage FlexVol®

```
trident_lun_pool_STORAGEPREFIX_RANDOMSTRING
```

- `STORAGEPREFIX` est la valeur de `storagePrefix` dans votre configuration backend. La valeur par défaut est `trident`.
- `RANDOMSTRING` est n'importe quelle chaîne de caractères que vous choisissez.

Exigence de dénomination des LUN

Le LUN doit être nommé `lun0`.

Exemple

Si votre `storagePrefix` est `xyz`, le chemin complet vers le LUN est :

```
trident_lun_pool_xyz_randomstring/lun0
```

Élément

Trident prend en charge le logiciel NetApp Element et l'importation de volumes NetApp HCI à l'aide du pilote `solidfire-san`.

REMARQUE	Le pilote Element prend en charge les noms de volumes identiques. Cependant, Trident renvoie une erreur s'il existe des noms de volumes en double. Pour contourner ce problème, clonez le volume, attribuez un nom de volume unique et importez le volume cloné.

L'exemple suivant importe un `element-managed` volume sur le backend `element_default`.

```
tridentctl import volume element_default element-managed -f pvc-basic-
import.yaml -n trident -d

+----------------------------------------+--------+--------------
+----------+----------------------------------------+--------+---------+
|                  NAME                  |  SIZE  | STORAGE CLASS |
PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+----------------------------------------+--------+--------------
+----------+----------------------------------------+--------+---------+
| pvc-970ce1ca-2096-4ecd-8545-ac7edc24a8fe | 10 GiB | basic-element |
block    | d3ba047a-ea0b-43f9-9c42-e38e58301c49 | online | true    |
+----------------------------------------+--------+--------------
+----------+----------------------------------------+--------+---------+
```

Azure NetApp Files

Trident prend en charge l'importation de volumes à l'aide du `azure-netapp-files` driver.

REMARQUE	Pour importer un volume Azure NetApp Files, identifiez le volume par son chemin de volume. Le chemin de volume est la partie du chemin d'exportation du volume après le `:/`. Par exemple, si le chemin de montage est `10.0.0.2:/importvol1`, le chemin de volume est `importvol1`.

L'exemple suivant importe un `azure-netapp-files` volume sur le backend `azurenetappfiles_40517` avec le chemin de volume `importvol1`.

```
tridentctl import volume azurenetappfiles_40517 importvol1 -f <path-to-
pvc-file> -n trident

+----------------------------------------+---------+--------------
+----------+----------------------------------------+--------+---------+
|                   NAME                 |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+----------------------------------------+---------+--------------
+----------+----------------------------------------+--------+---------+
| pvc-0ee95d60-fd5c-448d-b505-b72901b3a4ab | 100 GiB | anf-storage   |
file     | 1c01274f-d94b-44a3-98a3-04c953c9a51e | online | true    |
+----------------------------------------+---------+--------------
+----------+----------------------------------------+--------+---------+
```

Google Cloud NetApp Volumes

Trident prend en charge l'importation de volumes à l'aide du `google-cloud-netapp-volumes` driver.

L'exemple suivant importe un volume sur le backend `backend-tbc-gcnv1` avec le volume `testvoleasiaeast1`.

```
tridentctl import volume backend-tbc-gcnv1 "testvoleasiaeast1" -f < path-
to-pvc> -n trident

+----------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
|                   NAME                 |  SIZE   | STORAGE CLASS
| PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+----------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
| pvc-a69cda19-218c-4ca9-a941-aea05dd13dc0 |  10 GiB | gcnv-nfs-sc-
identity | file     | 8c18cdf1-0770-4bc0-bcc5-c6295fe6d837 | online | true
|
+----------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
```

L'exemple suivant importe un `google-cloud-netapp-volumes` volume lorsque deux volumes sont présents dans la même région :

```
tridentctl import volume backend-tbc-gcnv1
"projects/123456789100/locations/asia-east1-a/volumes/testvoleasiaeast1"
-f <path-to-pvc> -n trident

+------------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
|                   NAME                   |  SIZE   | STORAGE CLASS
| PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+------------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
| pvc-a69cda19-218c-4ca9-a941-aea05dd13dc0 |  10 GiB | gcnv-nfs-sc-
identity | file     | 8c18cdf1-0770-4bc0-bcc5-c6295fe6d837 | online | true
|
+------------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
```

Personnalisez les noms et les étiquettes des volumes

Avec Trident, vous pouvez attribuer des noms et des étiquettes explicites aux volumes que vous créez. Cela vous aide à identifier et à associer facilement les volumes à leurs ressources Kubernetes (PVC) respectives. Vous pouvez également définir des modèles au niveau du backend pour créer des noms et des étiquettes de volumes personnalisés ; tous les volumes que vous créez, importez ou clonez respecteront ces modèles.

Avant de commencer

Prise en charge des noms et étiquettes de volume personnalisables :

- Opérations de création, d'importation et de clonage de volume.
- Dans le cas du `ontap-nas-economy`driver, seul le nom du volume Qtree est conforme au modèle de nom.
- Dans le cas du `ontap-san-economy`driver, seul le nom du LUN est conforme au modèle de nom.

Limitations

- Les noms de volumes personnalisés sont compatibles uniquement avec les pilotes ONTAP sur site.
- Les étiquettes personnalisées ne sont prises en charge que pour les `ontap-san`, `ontap-nas`, et `ontap-nas-flexgroup` pilotes.
- Les noms de volumes personnalisés ne s'appliquent pas aux volumes existants.

Comportements clés des noms de volumes personnalisables

- En cas d'échec dû à une syntaxe incorrecte dans un modèle de nom, la création du backend échoue. Cependant, si l'application du modèle échoue, le volume sera nommé selon la convention de nommage

existante.

- Le préfixe de stockage n'est pas applicable lorsqu'un volume est nommé à l'aide d'un modèle de nom issu de la configuration du backend. Toute valeur de préfixe souhaitée peut être ajoutée directement au modèle.

Exemples de configuration backend avec modèle de nom et labels

Des modèles de noms personnalisés peuvent être définis au niveau racine et/ou au niveau du pool.

Exemple de niveau racine

```
{
  "version": 1,
  "storageDriverName": "ontap-nas",
  "backendName": "ontap-nfs-backend",
  "managementLIF": "<ip address>",
  "svm": "svm0",
  "username": "<admin>",
  "password": "<password>",
  "defaults": {
    "nameTemplate":
"{{.volume.Name}}_{{.labels.cluster}}_{{.volume.Namespace}}_{{.volume.Requ
estName}}"
  },
  "labels": {
    "cluster": "ClusterA",
    "PVC": "{{.volume.Namespace}}_{{.volume.RequestName}}"
  }
}
```

Exemple au niveau du pool

```
{
  "version": 1,
  "storageDriverName": "ontap-nas",
  "backendName": "ontap-nfs-backend",
  "managementLIF": "<ip address>",
  "svm": "svm0",
  "username": "<admin>",
  "password": "<password>",
  "useREST": true,
  "storage": [
    {
      "labels": {
        "labelname": "label1",
        "name": "{{ .volume.Name }}"
      },
      "defaults": {
        "nameTemplate": "pool01_{{ .volume.Name }}_{{ .labels.cluster
}}_{{ .volume.Namespace }}_{{ .volume.RequestName }}"
      }
    },
    {
      "labels": {
        "cluster": "label2",
        "name": "{{ .volume.Name }}"
      },
      "defaults": {
        "nameTemplate": "pool02_{{ .volume.Name }}_{{ .labels.cluster
}}_{{ .volume.Namespace }}_{{ .volume.RequestName }}"
      }
    }
  ]
}
```

Exemples de modèles de noms

Exemple 1:

```
"nameTemplate": "{{ .config.StoragePrefix }}_{{ .volume.Name }}_{{
.config.BackendName }}"
```

Exemple 2:

```
"nameTemplate": "pool_{{ .config.StoragePrefix }}_{{ .volume.Name }}_{{
slice .volume.RequestName 1 5 }}""
```

Points à considérer

1. Dans le cas des importations de volumes, les étiquettes ne sont mises à jour que si le volume existant possède des étiquettes dans un format spécifique. Par exemple : `{"provisioning":{"Cluster":"ClusterA", "PVC": "pvcname"}}`.
2. Dans le cas des importations de volumes gérés, le nom du volume suit le modèle de nom défini au niveau racine dans la définition du backend.
3. Trident ne prend pas en charge l'utilisation d'un opérateur de découpage avec le préfixe de stockage.
4. Si les modèles ne produisent pas de noms de volumes uniques, Trident ajoutera quelques caractères aléatoires pour créer des noms de volumes uniques.
5. Si le nom personnalisé d'un volume NAS economy dépasse 64 caractères, Trident nommera les volumes selon la convention de nommage existante. Pour tous les autres pilotes ONTAP, si le nom du volume dépasse la limite de caractères, le processus de création du volume échoue.

Partager un volume NFS entre espaces de noms

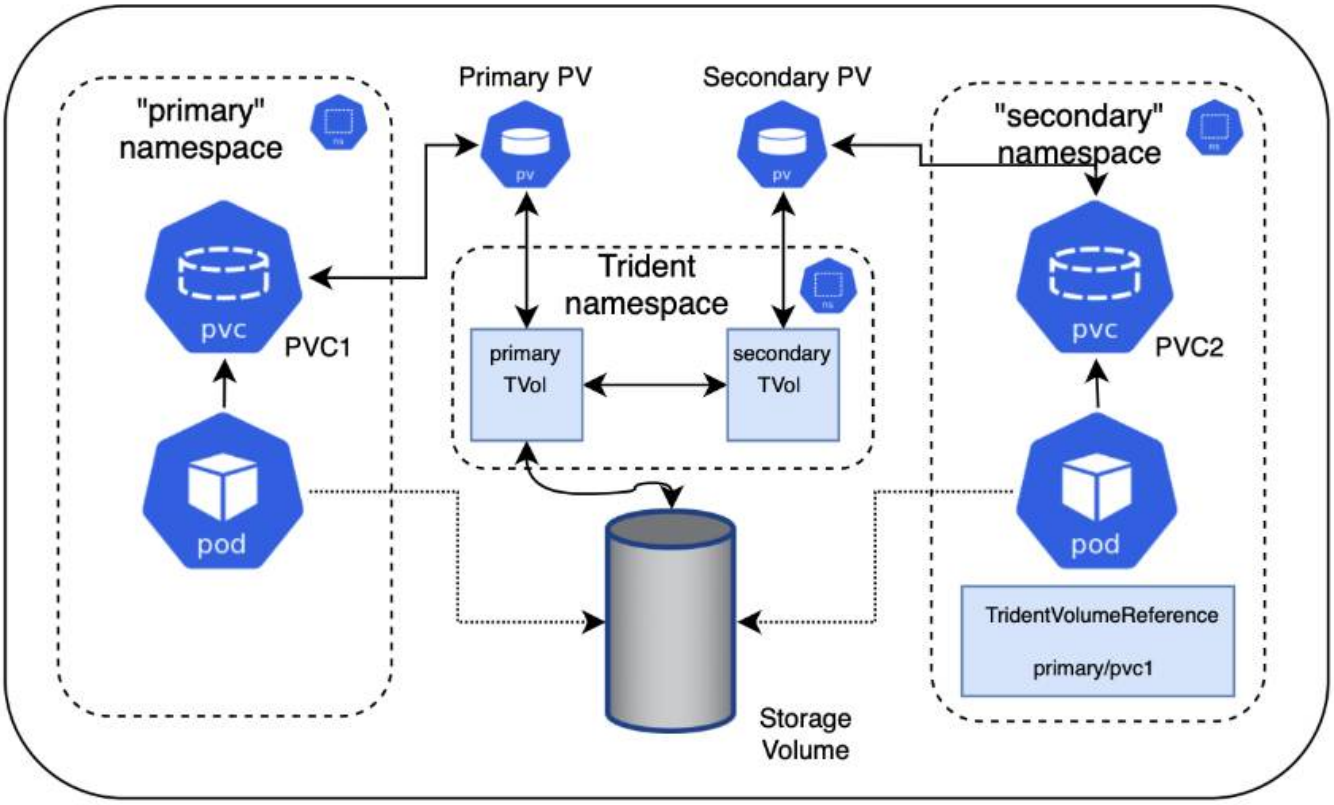
Avec Trident, vous pouvez créer un volume dans un espace de noms principal et le partager dans un ou plusieurs espaces de noms secondaires.

Caractéristiques

Le TridentVolumeReference CR vous permet de partager en toute sécurité des volumes NFS ReadWriteMany (RWX) entre un ou plusieurs espaces de noms Kubernetes. Cette solution native Kubernetes présente les avantages suivants :

- Plusieurs niveaux de contrôle d'accès pour garantir la sécurité
- Fonctionne avec tous les pilotes de volumes NFS Trident
- Aucune dépendance à tridentctl ni à toute autre fonctionnalité non native de Kubernetes

Ce diagramme illustre le partage de volumes NFS entre deux espaces de noms Kubernetes.



Démarrage rapide

Vous pouvez configurer le partage de volume NFS en quelques étapes seulement.

1 **Configurez le PVC source pour partager le volume**

Le propriétaire de l'espace de noms source accorde l'autorisation d'accéder aux données du PVC source.

2 **Autoriser la création d'un CR dans l'espace de noms de destination**

L'administrateur du cluster autorise le propriétaire de l'espace de noms de destination à créer la TridentVolumeReference CR.

3 **Créer TridentVolumeReference dans l'espace de noms de destination**

Le propriétaire de l'espace de noms de destination crée la TridentVolumeReference CR pour faire référence au PVC source.

4 **Créez le PVC subordonné dans l'espace de noms de destination**

Le propriétaire de l'espace de noms de destination crée le PVC subordonné pour utiliser la source de données du PVC source.

Configurez les espaces de noms source et de destination

Pour garantir la sécurité, le partage entre espaces de noms nécessite la collaboration et l'intervention du propriétaire de l'espace de noms source, de l'administrateur du cluster et du propriétaire de l'espace de noms de destination. Le rôle de l'utilisateur est défini à chaque étape.

Étapes

1. **Propriétaire de l'espace de noms source :** Créez le PVC (`pvc1` dans l'espace de noms source qui autorise le partage avec l'espace de noms de destination (`namespace2` en utilisant l'annotation `shareToNamespace`.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc1
  namespace: namespace1
  annotations:
    trident.netapp.io/shareToNamespace: namespace2
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: trident-csi
  resources:
    requests:
      storage: 100Gi
```

   Trident crée le PV et son volume de stockage NFS backend.

   REMARQUE

   - Vous pouvez partager le PVC avec plusieurs espaces de noms à l'aide d'une liste séparée par des virgules. Par exemple, `trident.netapp.io/shareToNamespace: namespace2,namespace3,namespace4`.
   - Vous pouvez partager avec tous les espaces de noms en utilisant `*`. Par exemple, `trident.netapp.io/shareToNamespace: *`
   - Vous pouvez mettre à jour le PVC pour inclure l'annotation `shareToNamespace` à tout moment.

2. **Administrateur du cluster :** Assurez-vous qu'un contrôle d'accès basé sur les rôles (RBAC) approprié est en place pour accorder au propriétaire de l'espace de noms de destination l'autorisation de créer la TridentVolumeReference CR dans l'espace de noms de destination.
3. **Propriétaire de l'espace de noms de destination :** Créez une CR TridentVolumeReference dans l'espace de noms de destination qui fait référence à l'espace de noms source `pvc1`.

```
apiVersion: trident.netapp.io/v1
kind: TridentVolumeReference
metadata:
  name: my-first-tvr
  namespace: namespace2
spec:
  pvcName: pvc1
  pvcNamespace: namespace1
```

4. **Propriétaire de l'espace de noms de destination :** Créez un PVC (`pvc2` dans l'espace de noms de destination (`namespace2` en utilisant l'annotation `shareFromPVC` pour désigner le PVC source.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  annotations:
    trident.netapp.io/shareFromPVC: namespace1/pvc1
  name: pvc2
  namespace: namespace2
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: trident-csi
  resources:
    requests:
      storage: 100Gi
```

REMARQUE La taille du PVC de destination doit être inférieure ou égale à celle du PVC source.

Résultats

Trident lit l' `shareFromPVC`annotation sur le PVC de destination et crée le PV de destination comme un volume subordonné sans ressource de stockage propre qui pointe vers le PV source et partage la ressource de stockage du PV source. Le PVC et le PV de destination apparaissent liés normalement.

Supprimer un volume partagé

Vous pouvez supprimer un volume partagé entre plusieurs espaces de noms. Trident supprimera l'accès au volume dans l'espace de noms source et maintiendra l'accès pour les autres espaces de noms qui partagent le volume. Lorsque tous les espaces de noms qui référencent le volume sont supprimés, Trident supprime le volume.

Utilisez `tridentctl get` pour interroger les volumes subordonnés

À l'aide de l'[`tridentctl`utilitaire, vous pouvez exécuter la commande `get`pour obtenir les volumes subordonnés. Pour plus d'informations, consultez le lien :

../trident-reference/trident-cl.html[`tridentctl commandes et options].

```
Usage:
  tridentctl get [option]
```

Drapeaux:

- `-h, --help`: Aide pour les volumes.
- `--parentOfSubordinate string` : Limiter la requête au volume source subordonné.
- `--subordinateOf string` : Limiter la requête aux subordonnés du volume.

Limitations

- Trident ne peut pas empêcher les espaces de noms de destination d'écrire sur le volume partagé. Vous devez utiliser le verrouillage de fichiers ou d'autres processus pour éviter l'écrasement des données du volume partagé.
- Vous ne pouvez pas révoquer l'accès au PVC source en supprimant les `shareToNamespace` ou `shareFromNamespace` annotations ou en supprimant le `TridentVolumeReference` CR. Pour révoquer l'accès, vous devez supprimer le PVC subordonné.
- Les instantanés, les clones et la mise en miroir ne sont pas possibles sur les volumes subordonnés.

Pour plus d'informations

Pour en savoir plus sur l'accès aux volumes entre espaces de noms :

- Regardez la démo sur "NetAppTV".

Cloner des volumes entre espaces de noms

Avec Trident, vous pouvez créer de nouveaux volumes à l'aide de volumes existants ou de volumesnapshots provenant d'un espace de noms différent au sein du même cluster Kubernetes.

Prérequis

Avant de cloner des volumes, assurez-vous que les backends source et de destination sont du même type et ont la même classe de stockage.

REMARQUE	Le clonage entre espaces de noms est pris en charge uniquement pour les `ontap-san` et `ontap-nas` pilotes de stockage. Les clones en lecture seule ne sont pas pris en charge.

Démarrage rapide

Vous pouvez configurer le clonage de volume en quelques étapes seulement.

1. **Configurez le PVC source pour cloner le volume**

Le propriétaire de l'espace de noms source accorde l'autorisation d'accéder aux données du PVC source.

2 Autoriser la création d'un CR dans l'espace de noms de destination

L'administrateur du cluster autorise le propriétaire de l'espace de noms de destination à créer la TridentVolumeReference CR.

3 Créer TridentVolumeReference dans l'espace de noms de destination

Le propriétaire de l'espace de noms de destination crée la TridentVolumeReference CR pour faire référence au PVC source.

4 Créez le PVC cloné dans l'espace de noms de destination

Le propriétaire de l'espace de noms de destination crée un PVC pour cloner le PVC de l'espace de noms source.

Configurez les espaces de noms source et de destination

Pour garantir la sécurité, le clonage de volumes entre espaces de noms nécessite la collaboration et l'intervention du propriétaire de l'espace de noms source, de l'administrateur du cluster et du propriétaire de l'espace de noms de destination. Le rôle de l'utilisateur est défini à chaque étape.

Étapes

1. **Propriétaire de l'espace de noms source :** Créez le PVC (`pvc1`) dans l'espace de noms source (`namespace1`) qui autorise le partage avec l'espace de noms de destination (`namespace2`) en utilisant l'annotation `cloneToNamespace`.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc1
  namespace: namespace1
  annotations:
    trident.netapp.io/cloneToNamespace: namespace2
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: trident-csi
  resources:
    requests:
      storage: 100Gi
```

   Trident crée le PV et son volume de stockage backend.

REMARQUE

- Vous pouvez partager le PVC avec plusieurs espaces de noms à l'aide d'une liste séparée par des virgules. Par exemple, `trident.netapp.io/cloneToNamespace: namespace2,namespace3,namespace4`.
- Vous pouvez partager avec tous les espaces de noms en utilisant `*`. Par exemple, `trident.netapp.io/cloneToNamespace: *`
- Vous pouvez mettre à jour le PVC pour inclure l'annotation `cloneToNamespace` à tout moment.

2. **Administrateur du cluster :** Assurez-vous que le RBAC approprié est en place pour accorder l'autorisation au propriétaire de l'espace de noms de destination de créer la TridentVolumeReference CR dans l'espace de noms de destination (`namespace2`).
3. **Propriétaire de l'espace de noms de destination :** Créez une CR TridentVolumeReference dans l'espace de noms de destination qui fait référence à l'espace de noms source `pvc1`.

```
apiVersion: trident.netapp.io/v1
kind: TridentVolumeReference
metadata:
  name: my-first-tvr
  namespace: namespace2
spec:
  pvcName: pvc1
  pvcNamespace: namespace1
```

4. **Propriétaire de l'espace de noms de destination :** Créez un PVC (`pvc2`) dans l'espace de noms de destination (`namespace2`) en utilisant les `cloneFromPVC` ou `cloneFromSnapshot`, et `cloneFromNamespace` annotations pour désigner le PVC source.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  annotations:
    trident.netapp.io/cloneFromPVC: pvc1
    trident.netapp.io/cloneFromNamespace: namespace1
  name: pvc2
  namespace: namespace2
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: trident-csi
  resources:
    requests:
      storage: 100Gi
```

Limitations

- Pour les PVC provisionnés à l'aide des pilotes ontap-nas-economy, les clones en lecture seule ne sont pas pris en charge.

Répliquer les volumes à l'aide de SnapMirror

Trident prend en charge les relations de miroir entre un volume source sur un cluster et le volume de destination sur le cluster homologue pour la réplication des données en vue de la reprise après sinistre. Vous pouvez utiliser une définition de ressource personnalisée (CRD) avec espace de noms, appelée Trident Mirror Relationship (TMR), pour effectuer les opérations suivantes :

- Créer des relations de miroir entre les volumes (PVCs)
- Supprimer les relations de miroir entre les volumes
- Briser les relations miroir
- Promouvoir le volume secondaire pendant des conditions de sinistre (basculements)
- Effectuer une transition sans perte des applications d'un cluster à l'autre (lors de basculements ou de migrations planifiés)

Prérequis de réplication

Assurez-vous que les conditions préalables suivantes sont remplies avant de commencer :

clusters ONTAP

- **Trident** : la version 22.10 ou ultérieure de Trident doit être présente sur les clusters Kubernetes source et de destination qui utilisent ONTAP comme backend.
- **Licences** : Les licences asynchrones ONTAP SnapMirror utilisant le Data Protection bundle doivent être activées sur les clusters ONTAP source et de destination. Consultez "SnapMirror aperçu des licences dans ONTAP" pour plus d'informations.

  À compter de ONTAP 9.10.1, toutes les licences sont fournies sous forme de fichier de licence NetApp (NLF), qui est un fichier unique permettant d'activer plusieurs fonctionnalités. Consultez "Licences incluses avec ONTAP One" pour plus d'informations.

REMARQUE	Seule la protection asynchrone SnapMirror est prise en charge.

Interconnexion

- **Cluster et SVM** : Les backends de stockage ONTAP doivent être appariés. Consultez "Aperçu du peering de cluster et de SVM" pour plus d'informations.

IMPORTANT	Assurez-vous que les noms SVM utilisés dans la relation de réplication entre deux clusters ONTAP sont uniques.

- **Trident et SVM** : les SVM distants appariés doivent être disponibles pour Trident sur le cluster de destination.

Pilotes pris en charge

NetApp Trident prend en charge la réplication de volumes avec la technologie NetApp SnapMirror en utilisant

des classes de stockage prises en charge par les pilotes suivants : **`ontap-nas`: NFS** `ontap-san`: iSCSI **`ontap-san`: FC** `ontap-san`: NVMe/TCP (nécessite la version ONTAP 9.15.1 au minimum)

REMARQUE	La réplication de volumes à l'aide de SnapMirror n'est pas prise en charge pour les systèmes ASA r2. Pour des informations sur les systèmes ASA r2, voir "En savoir plus sur les systèmes de stockage ASA r2".

Créer un PVC en miroir

Suivez ces étapes et utilisez les exemples CRD pour créer une relation miroir entre les volumes primaires et secondaires.

Étapes

1. Effectuez les étapes suivantes sur le cluster Kubernetes principal :
   a. Créez un objet StorageClass avec le paramètre `trident.netapp.io/replication: true`.

      Exemple

      ```
      apiVersion: storage.k8s.io/v1
      kind: StorageClass
      metadata:
        name: csi-nas
      provisioner: csi.trident.netapp.io
      parameters:
        backendType: "ontap-nas"
        fsType: "nfs"
        trident.netapp.io/replication: "true"
      ```

   b. Créez un PVC avec le StorageClass précédemment créé.

      Exemple

      ```
      kind: PersistentVolumeClaim
      apiVersion: v1
      metadata:
        name: csi-nas
      spec:
        accessModes:
        - ReadWriteMany
        resources:
          requests:
            storage: 1Gi
        storageClassName: csi-nas
      ```

   c. Créez un CR MirrorRelationship avec des informations locales.

Exemple

```
kind: TridentMirrorRelationship
apiVersion: trident.netapp.io/v1
metadata:
  name: csi-nas
spec:
  state: promoted
  volumeMappings:
  - localPVCName: csi-nas
```

Trident récupère les informations internes du volume et l'état actuel de protection des données (DP) du volume, puis remplit le champ d'état du MirrorRelationship.

d. Obtenez le TridentMirrorRelationship CR pour obtenir le nom interne et le SVM du PVC.

```
kubectl get tmr csi-nas
```

```
kind: TridentMirrorRelationship
apiVersion: trident.netapp.io/v1
metadata:
  name: csi-nas
  generation: 1
spec:
  state: promoted
  volumeMappings:
  - localPVCName: csi-nas
status:
  conditions:
  - state: promoted
    localVolumeHandle:
"datavserver:trident_pvc_3bedd23c_46a8_4384_b12b_3c38b313c1e1"
    localPVCName: csi-nas
    observedGeneration: 1
```

2. Effectuez les étapes suivantes sur le cluster Kubernetes secondaire :

   a. Créez un StorageClass avec le paramètre trident.netapp.io/replication: true.

Exemple

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: csi-nas
provisioner: csi.trident.netapp.io
parameters:
  trident.netapp.io/replication: true
```

b. Créez un CR MirrorRelationship avec les informations de destination et de source.

Exemple

```
kind: TridentMirrorRelationship
apiVersion: trident.netapp.io/v1
metadata:
  name: csi-nas
spec:
  state: established
  volumeMappings:
  - localPVCName: csi-nas
    remoteVolumeHandle:
"datavserver:trident_pvc_3bedd23c_46a8_4384_b12b_3c38b313c1e1"
```

Trident créera une relation SnapMirror avec le nom de stratégie de relation configuré (ou par défaut pour ONTAP) et l'initialisera.

c. Créez un PVC avec la StorageClass précédemment créée pour servir de SnapMirror destination secondaire.

Exemple

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: csi-nas
  annotations:
    trident.netapp.io/mirrorRelationship: csi-nas
spec:
  accessModes:
  - ReadWriteMany
resources:
  requests:
    storage: 1Gi
storageClassName: csi-nas
```

Trident vérifiera la présence de la TridentMirrorRelationship CRD et ne créera pas le volume si la relation n'existe pas. Si la relation existe, Trident s'assurera que le nouveau volume FlexVol est placé sur une SVM appairée avec la SVM distante définie dans la MirrorRelationship.

États de réplication du volume

Une Trident Mirror Relationship (TMR) est une CRD qui représente une extrémité d'une relation de réplication entre des PVC. La TMR de destination possède un état, qui indique à Trident quel est l'état souhaité. La TMR de destination possède les états suivants :

- **Établi** : le PVC local est le volume de destination d'une relation miroir, et il s'agit d'une nouvelle relation.
- **Promu** : le PVC local est ReadWrite et montable, sans relation miroir actuellement en vigueur.
- **Rétabli** : le PVC local est le volume de destination d'une relation miroir et était également auparavant dans cette relation miroir.
  - L'état rétabli doit être utilisé si le volume de destination a déjà été en relation avec le volume source, car il écrase le contenu du volume de destination.
  - L'état rétabli échouera si le volume n'était pas auparavant en relation avec la source.

Promouvoir le PVC secondaire lors d'un basculement imprévu

Effectuez l'étape suivante sur le cluster Kubernetes secondaire :

- Mettez à jour le champ *spec.state* de TridentMirrorRelationship à `promoted`.

Promouvoir le PVC secondaire lors d'un basculement planifié

Lors d'un basculement (migration) planifié, effectuez les étapes suivantes pour promouvoir le PVC secondaire :

Étapes

1. Sur le cluster Kubernetes principal, créez un instantané du PVC et attendez que l'instantané soit créé.
2. Sur le cluster Kubernetes principal, créez le CR SnapshotInfo pour obtenir les détails internes.

   Exemple

   ```
   kind: SnapshotInfo
   apiVersion: trident.netapp.io/v1
   metadata:
     name: csi-nas
   spec:
     snapshot-name: csi-nas-snapshot
   ```

3. Sur le cluster Kubernetes secondaire, mettez à jour le champ *spec.state* de la ressource personnalisée *TridentMirrorRelationship* à *promoted* et *spec.promotedSnapshotHandle* pour qu'il soit le internalName de l'instantané.
4. Sur le cluster Kubernetes secondaire, confirmez l'état (champ status.state) de TridentMirrorRelationship sur promu.

Rétablir une relation miroir après un basculement

Avant de rétablir une relation miroir, choisissez le côté que vous souhaitez définir comme nouveau principal.

Étapes

1. Sur le cluster Kubernetes secondaire, assurez-vous que les valeurs du champ *spec.remoteVolumeHandle* sur le TridentMirrorRelationship sont mises à jour.
2. Sur le cluster Kubernetes secondaire, mettez à jour le champ *spec.mirror* de TridentMirrorRelationship à `reestablished`.

Opérations supplémentaires

Trident prend en charge les opérations suivantes sur les volumes primaires et secondaires :

Répliquer le PVC primaire vers un nouveau PVC secondaire

Assurez-vous de disposer déjà d'un PVC primaire et d'un PVC secondaire.

Étapes

1. Supprimez les CRD PersistentVolumeClaim et TridentMirrorRelationship du cluster secondaire (de destination) établi.
2. Supprimez le CRD TridentMirrorRelationship du cluster principal (source).
3. Créez un nouveau TridentMirrorRelationship CRD sur le cluster principal (source) pour le nouveau PVC secondaire (destination) que vous souhaitez établir.

Redimensionner un PVC miroir, principal ou secondaire

Le PVC peut être redimensionné normalement, ONTAP étendra automatiquement les flevxols de destination si la quantité de données dépasse la taille actuelle.

Supprimer la réplication d'un PVC

Pour supprimer la réplication, effectuez l'une des opérations suivantes sur le volume secondaire actuel :

- Supprimez le MirrorRelationship sur le PVC secondaire. Cela interrompt la relation de réplication.
- Ou, mettez à jour le champ spec.state à *promoted*.

Supprimer un PVC (qui était précédemment mis en miroir)

Trident vérifie la présence de PVC répliqués et libère la relation de réplication avant de tenter de supprimer le volume.

Supprimer un TMR

La suppression d'un TMR sur l'un des côtés d'une relation miroir entraîne la transition du TMR restant à l'état *promoted* avant que Trident ne finalise la suppression. Si le TMR sélectionné pour suppression est déjà à l'état *promoted*, il n'existe aucune relation miroir et le TMR sera supprimé et Trident promouvra le PVC local à *ReadWrite*. Cette suppression libère les métadonnées SnapMirror pour le volume local dans ONTAP. Si ce volume est utilisé dans une relation miroir à l'avenir, il devra utiliser un nouveau TMR avec un état de réplication de volume *established* lors de la création de la nouvelle relation miroir.

Mettez à jour les relations miroir lorsque ONTAP est en ligne

Les relations de miroir peuvent être mises à jour à tout moment après leur établissement. Vous pouvez utiliser le `state: promoted` ou `state: reestablished` champ pour mettre à jour les relations. Lors de la promotion d'un volume de destination vers un volume ReadWrite standard, vous pouvez utiliser *promotedSnapshotHandle* pour spécifier un instantané précis auquel restaurer le volume actuel.

Mettre à jour les relations miroir lorsque ONTAP est hors ligne

Vous pouvez utiliser une CRD pour effectuer une mise à jour SnapMirror sans que Trident ait de connexion directe au cluster ONTAP. Consultez l'exemple de format suivant de TridentActionMirrorUpdate :

Exemple

```
apiVersion: trident.netapp.io/v1
kind: TridentActionMirrorUpdate
metadata:
  name: update-mirror-b
spec:
  snapshotHandle: "pvc-1234/snapshot-1234"
  tridentMirrorRelationshipName: mirror-b
```

`status.state` reflète l'état de la TridentActionMirrorUpdate CRD. Elle peut prendre une valeur parmi *Succeeded*, *In Progress* ou *Failed*.

Utiliser la topologie CSI

Trident peut créer et attacher sélectivement des volumes aux nœuds présents dans un cluster Kubernetes en utilisant le "Fonctionnalité de topologie CSI".

Aperçu

Grâce à la fonctionnalité CSI Topology, l'accès aux volumes peut être limité à un sous-ensemble de nœuds, en fonction des régions et des zones de disponibilité. Les fournisseurs de cloud permettent aujourd'hui aux administrateurs Kubernetes de déployer des nœuds basés sur les zones. Les nœuds peuvent être situés dans différentes zones de disponibilité au sein d'une région, ou répartis sur diverses régions. Pour faciliter le provisionnement des volumes pour les charges de travail dans une architecture multizone, Trident utilise CSI Topology.

ASTUCE	Apprenez-en davantage sur la fonctionnalité de topologie CSI "ici".

Kubernetes propose deux modes de liaison de volumes uniques :

- Avec `VolumeBindingMode` défini sur `Immediate`, Trident crée le volume sans aucune prise en compte de la topologie. La liaison de volume et le provisionnement dynamique sont gérés lors de la création du PVC. C'est le comportement par défaut `VolumeBindingMode` et il convient aux clusters qui n'imposent pas de contraintes de topologie. Les volumes persistants sont créés sans dépendre des exigences de planification du pod demandeur.
- Avec `VolumeBindingMode` défini sur `WaitForFirstConsumer`, la création et la liaison d'un volume persistant pour un PVC sont différées jusqu'à ce qu'un pod utilisant ce PVC soit planifié et créé. De cette façon, les volumes sont créés pour répondre aux contraintes de planification imposées par les exigences

de topologie.

REMARQUE	Le `WaitForFirstConsumer` mode de liaison ne nécessite pas d'étiquettes de topologie. Cela peut être utilisé indépendamment de la fonctionnalité de topologie CSI.

Ce dont vous aurez besoin

Pour utiliser la topologie CSI, vous avez besoin des éléments suivants :

- Un cluster Kubernetes exécutant un "Version Kubernetes prise en charge"

```
kubectl version
Client Version: version.Info{Major:"1", Minor:"19",
GitVersion:"v1.19.3",
GitCommit:"1e11e4a2108024935ecfcb2912226cedeafd99df",
GitTreeState:"clean", BuildDate:"2020-10-14T12:50:19Z",
GoVersion:"go1.15.2", Compiler:"gc", Platform:"linux/amd64"}
Server Version: version.Info{Major:"1", Minor:"19",
GitVersion:"v1.19.3",
GitCommit:"1e11e4a2108024935ecfcb2912226cedeafd99df",
GitTreeState:"clean", BuildDate:"2020-10-14T12:41:49Z",
GoVersion:"go1.15.2", Compiler:"gc", Platform:"linux/amd64"}
```

- Les nœuds du cluster doivent comporter des étiquettes permettant de prendre en compte la topologie (`topology.kubernetes.io/region` et `topology.kubernetes.io/zone`). Ces étiquettes **doivent être présentes sur les nœuds du cluster** avant l'installation de Trident pour que Trident puisse prendre en compte la topologie.

```
kubectl get nodes -o=jsonpath='{range .items[*]}[{.metadata.name},
{.metadata.labels}]{"\n"}{end}' | grep --color "topology.kubernetes.io"
[node1,
{"beta.kubernetes.io/arch":"amd64","beta.kubernetes.io/os":"linux","kube
rnetes.io/arch":"amd64","kubernetes.io/hostname":"node1","kubernetes.io/
os":"linux","node-
role.kubernetes.io/master":"","topology.kubernetes.io/region":"us-
east1","topology.kubernetes.io/zone":"us-east1-a"}]
[node2,
{"beta.kubernetes.io/arch":"amd64","beta.kubernetes.io/os":"linux","kube
rnetes.io/arch":"amd64","kubernetes.io/hostname":"node2","kubernetes.io/
os":"linux","node-
role.kubernetes.io/worker":"","topology.kubernetes.io/region":"us-
east1","topology.kubernetes.io/zone":"us-east1-b"}]
[node3,
{"beta.kubernetes.io/arch":"amd64","beta.kubernetes.io/os":"linux","kube
rnetes.io/arch":"amd64","kubernetes.io/hostname":"node3","kubernetes.io/
os":"linux","node-
role.kubernetes.io/worker":"","topology.kubernetes.io/region":"us-
east1","topology.kubernetes.io/zone":"us-east1-c"}]
```

Étape 1 : Créez un backend tenant compte de la topologie

Les backends de stockage Trident peuvent être conçus pour provisionner sélectivement des volumes en fonction des zones de disponibilité. Chaque backend peut comporter un bloc `supportedTopologies` optionnel qui représente une liste de zones et de régions prises en charge. Pour les StorageClasses qui utilisent un tel backend, un volume ne serait créé que si la demande provient d'une application planifiée dans une région ou une zone prise en charge.

Voici un exemple de définition de backend :

YAML

```yaml
---
version: 1
storageDriverName: ontap-san
backendName: san-backend-us-east1
managementLIF: 192.168.27.5
svm: iscsi_svm
username: admin
password: password
supportedTopologies:
  - topology.kubernetes.io/region: us-east1
    topology.kubernetes.io/zone: us-east1-a
  - topology.kubernetes.io/region: us-east1
    topology.kubernetes.io/zone: us-east1-b
```

JSON

```json
{
  "version": 1,
  "storageDriverName": "ontap-san",
  "backendName": "san-backend-us-east1",
  "managementLIF": "192.168.27.5",
  "svm": "iscsi_svm",
  "username": "admin",
  "password": "password",
  "supportedTopologies": [
    {
      "topology.kubernetes.io/region": "us-east1",
      "topology.kubernetes.io/zone": "us-east1-a"
    },
    {
      "topology.kubernetes.io/region": "us-east1",
      "topology.kubernetes.io/zone": "us-east1-b"
    }
  ]
}
```

REMARQUE

`supportedTopologies` est utilisé pour fournir une liste de régions et de zones par backend. Ces régions et zones représentent la liste des valeurs autorisées qui peuvent être fournies dans un StorageClass. Pour les StorageClasses qui contiennent un sous-ensemble des régions et zones fournies dans un backend, Trident crée un volume sur le backend.

Vous pouvez définir `supportedTopologies` par pool de stockage également. Voir l'exemple suivant :

```
---
version: 1
storageDriverName: ontap-nas
backendName: nas-backend-us-central1
managementLIF: 172.16.238.5
svm: nfs_svm
username: admin
password: password
supportedTopologies:
  - topology.kubernetes.io/region: us-central1
    topology.kubernetes.io/zone: us-central1-a
  - topology.kubernetes.io/region: us-central1
    topology.kubernetes.io/zone: us-central1-b
storage:
  - labels:
      workload: production
    supportedTopologies:
      - topology.kubernetes.io/region: us-central1
        topology.kubernetes.io/zone: us-central1-a
  - labels:
      workload: dev
    supportedTopologies:
      - topology.kubernetes.io/region: us-central1
        topology.kubernetes.io/zone: us-central1-b
```

Dans cet exemple, les `region` et `zone` étiquettes représentent l'emplacement du pool de stockage. `topology.kubernetes.io/region` et `topology.kubernetes.io/zone` indiquent où les pools de stockage peuvent être utilisés.

Étape 2 : Définir les StorageClasses qui sont conscients de la topologie

En fonction des étiquettes de topologie fournies aux nœuds du cluster, il est possible de définir des StorageClasses contenant des informations de topologie. Cela déterminera les pools de stockage qui servent de candidats pour les requêtes PVC effectuées, ainsi que le sous-ensemble de nœuds pouvant utiliser les volumes provisionnés par Trident.

Voir l'exemple suivant :

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: netapp-san-us-east1
provisioner: csi.trident.netapp.io
volumeBindingMode: WaitForFirstConsumer
allowedTopologies:
  - matchLabelExpressions:
    - key: topology.kubernetes.io/zone
      values:
        - us-east1-a
        - us-east1-b
    - key: topology.kubernetes.io/region
      values:
        - us-east1
parameters:
  fsType: ext4
```

Dans la définition StorageClass fournie ci-dessus, `volumeBindingMode` est définie sur `WaitForFirstConsumer`. Les PVC demandés avec cette StorageClass ne seront pas traités tant qu'ils ne seront pas référencés dans un pod. Et, `allowedTopologies` fournit les zones et la région à utiliser. La `netapp-san-us-east1` StorageClass crée des PVC sur le backend `san-backend-us-east1` défini ci-dessus.

Étape 3 : Créer et utiliser un PVC

Une fois le StorageClass créé et associé à un backend, vous pouvez désormais créer des PVC.

Voir l'exemple `spec` ci-dessous :

```
---
kind: PersistentVolumeClaim
apiVersion: v1
metadata: null
name: pvc-san
spec: null
accessModes:
  - ReadWriteOnce
resources:
  requests:
    storage: 300Mi
storageClassName: netapp-san-us-east1
```

La création d'un PVC à l'aide de ce manifeste donnerait le résultat suivant :

```
kubectl create -f pvc.yaml
persistentvolumeclaim/pvc-san created
kubectl get pvc
NAME      STATUS    VOLUME   CAPACITY   ACCESS MODES   STORAGECLASS
AGE
pvc-san   Pending                                      netapp-san-us-east1
2s
kubectl describe pvc
Name:          pvc-san
Namespace:     default
StorageClass:  netapp-san-us-east1
Status:        Pending
Volume:
Labels:        <none>
Annotations:   <none>
Finalizers:    [kubernetes.io/pvc-protection]
Capacity:
Access Modes:
VolumeMode:    Filesystem
Mounted By:    <none>
Events:
  Type    Reason                Age   From                         Message
  ----    ------                ----  ----                         -------
  Normal  WaitForFirstConsumer  6s    persistentvolume-controller  waiting
for first consumer to be created before binding
```

Pour que Trident puisse créer un volume et le lier au PVC, utilisez le PVC dans un pod. Voir l'exemple suivant :

```
apiVersion: v1
kind: Pod
metadata:
  name: app-pod-1
spec:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: topology.kubernetes.io/region
            operator: In
            values:
            - us-east1
      preferredDuringSchedulingIgnoredDuringExecution:
      - weight: 1
        preference:
          matchExpressions:
          - key: topology.kubernetes.io/zone
            operator: In
            values:
            - us-east1-a
            - us-east1-b
  securityContext:
    runAsUser: 1000
    runAsGroup: 3000
    fsGroup: 2000
  volumes:
  - name: vol1
    persistentVolumeClaim:
      claimName: pvc-san
  containers:
  - name: sec-ctx-demo
    image: busybox
    command: [ "sh", "-c", "sleep 1h" ]
    volumeMounts:
    - name: vol1
      mountPath: /data/demo
    securityContext:
      allowPrivilegeEscalation: false
```

Ce podSpec indique à Kubernetes de planifier le pod sur les nœuds présents dans la `us-east1` région, et de choisir parmi tous les nœuds présents dans la `us-east1-a` ou `us-east1-b` zones.

Voir la sortie suivante :

```
kubectl get pods -o wide
NAME        READY   STATUS    RESTARTS   AGE   IP               NODE
NOMINATED NODE   READINESS GATES
app-pod-1   1/1     Running   0          19s   192.168.25.131   node2
<none>           <none>
kubectl get pvc -o wide
NAME      STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS          AGE   VOLUMEMODE
pvc-san   Bound    pvc-ecb1e1a0-840c-463b-8b65-b3d033e2e62b   300Mi
RWO            netapp-san-us-east1   48s   Filesystem
```

Mettre à jour les backends pour inclure `supportedTopologies`

Les backends préexistants peuvent être mis à jour pour inclure une liste de `supportedTopologies` en utilisant `tridentctl backend update`. Cela n'affectera pas les volumes déjà provisionnés et ne sera utilisé que pour les PVC ultérieurs.

Pour plus d'informations

- "Gérer les ressources pour les conteneurs"
- "nodeSelector"
- "Affinité et anti-affinité"
- "Souillures et tolérances"

Travailler avec des snapshots

Les snapshots de volumes persistants (PV) Kubernetes permettent de créer des copies à un instant précis des volumes. Vous pouvez créer un snapshot d'un volume créé avec Trident, importer un snapshot créé en dehors de Trident, créer un nouveau volume à partir d'un snapshot existant, et récupérer des données de volume à partir de snapshots.

Aperçu

La capture instantanée de volume est prise en charge par `ontap-nas`, `ontap-nas-flexgroup`, `ontap-san`, `ontap-san-economy`, `solidfire-san`, `azure-netapp-files` et `google-cloud-netapp-volumes` pilotes.

Avant de commencer

Vous devez disposer d'un contrôleur de snapshots externe et de Custom Resource Definitions (CRDs) pour travailler avec les snapshots. C'est la responsabilité de l'orchestrateur Kubernetes (par exemple : Kubeadm, GKE, OpenShift).

Si votre distribution Kubernetes n'inclut pas le contrôleur de snapshots et les CRD, reportez-vous à Déployer un contrôleur d'instantané de volume.

REMARQUE	Ne créez pas de contrôleur de snapshots si vous créez des snapshots de volumes à la demande dans un environnement GKE. GKE utilise un contrôleur de snapshots intégré et masqué.

Créer un instantané de volume

Étapes

1. Créez un `VolumeSnapshotClass`. Pour plus d'informations, consultez "VolumeSnapshotClass".
   - Le `driver` pointe vers le pilote Trident CSI.
   - `deletionPolicy` peut être `Delete` ou `Retain`. Lorsqu'il est défini sur `Retain`, l'instantané physique sous-jacent sur le cluster de stockage est conservé même lorsque l'objet `VolumeSnapshot` est supprimé.

     Exemple

     ```
     cat snap-sc.yaml
     ```

     ```
     apiVersion: snapshot.storage.k8s.io/v1
     kind: VolumeSnapshotClass
     metadata:
       name: csi-snapclass
     driver: csi.trident.netapp.io
     deletionPolicy: Delete
     ```

2. Créer un instantané d'un PVC existant.

   Exemples

   - Cet exemple crée un instantané d'un PVC existant.

     ```
     cat snap.yaml
     ```

     ```
     apiVersion: snapshot.storage.k8s.io/v1
     kind: VolumeSnapshot
     metadata:
       name: pvc1-snap
     spec:
       volumeSnapshotClassName: csi-snapclass
       source:
         persistentVolumeClaimName: pvc1
     ```

   - Cet exemple crée un objet d'instantané de volume pour un PVC nommé `pvc1` et le nom de l'instantané est défini sur `pvc1-snap`. Un VolumeSnapshot est analogue à un PVC et est associé à un `VolumeSnapshotContent` objet qui représente l'instantané réel.

```
kubectl create -f snap.yaml
volumesnapshot.snapshot.storage.k8s.io/pvc1-snap created

kubectl get volumesnapshots
NAME                    AGE
pvc1-snap               50s
```

- Vous pouvez identifier l'`VolumeSnapshotContent`objet pour le `pvc1-snap` VolumeSnapshot en le décrivant. Le `Snapshot Content Name` identifie l'objet VolumeSnapshotContent qui sert cet instantané. Le `Ready To Use` paramètre indique que l'instantané peut être utilisé pour créer un nouveau PVC.

```
kubectl describe volumesnapshots pvc1-snap
Name:         pvc1-snap
Namespace:    default
...
Spec:
  Snapshot Class Name:    pvc1-snap
  Snapshot Content Name:  snapcontent-e8d8a0ca-9826-11e9-9807-
525400f3f660
  Source:
    API Group:
    Kind:       PersistentVolumeClaim
    Name:       pvc1
Status:
  Creation Time:  2019-06-26T15:27:29Z
  Ready To Use:   true
  Restore Size:   3Gi
...
```

Créer un PVC à partir d'un instantané de volume

Vous pouvez utiliser `dataSource` pour créer un PVC en utilisant un VolumeSnapshot nommé `<pvc-name>` comme source des données. Après la création du PVC, il peut être attaché à un pod et utilisé comme n'importe quel autre PVC.

AVERTISSEMENT	Le PVC sera créé dans le même backend que le volume source. Consultez "KB : La création d'un PVC à partir d'un instantané de PVC Trident ne peut pas être effectuée dans un autre backend".

L'exemple suivant crée le PVC en utilisant `pvc1-snap` comme source de données.

```
cat pvc-from-snap.yaml
```

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-from-snap
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: golden
  resources:
    requests:
      storage: 3Gi
  dataSource:
    name: pvc1-snap
    kind: VolumeSnapshot
    apiGroup: snapshot.storage.k8s.io
```

Importer un instantané de volume

Trident prend en charge la "Processus de snapshot pré-provisionné Kubernetes" pour permettre à l'administrateur du cluster de créer un `VolumeSnapshotContent` objet et d'importer des instantanés créés en dehors de Trident.

Avant de commencer

Trident doit avoir créé ou importé le volume parent de la capture.

Étapes

1. **Administrateur du cluster :** Créez un `VolumeSnapshotContent` objet qui référence l'instantané du backend. Cela lance le workflow d'instantané dans Trident.
   - Spécifiez le nom de l'instantané du backend dans `annotations` comme `trident.netapp.io/internalSnapshotName: <"backend-snapshot-name">`.
   - Spécifiez `<name-of-parent-volume-in-trident>/<volume-snapshot-content-name>` dans `snapshotHandle`. Il s'agit de la seule information fournie à Trident par le générateur de snapshots externe dans l'appel `ListSnapshots`.

REMARQUE	Le `<volumeSnapshotContentName>` ne peut pas toujours correspondre au nom de l'instantané du backend en raison des contraintes de dénomination des CR.

     Exemple

     L'exemple suivant crée un `VolumeSnapshotContent` objet qui référence l'instantané du backend `snap-01`.

```
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshotContent
metadata:
  name: import-snap-content
  annotations:
    trident.netapp.io/internalSnapshotName: "snap-01"  # This is the
name of the snapshot on the backend
spec:
  deletionPolicy: Retain
  driver: csi.trident.netapp.io
  source:
    snapshotHandle: pvc-f71223b5-23b9-4235-bbfe-e269ac7b84b0/import-
snap-content # <import PV name or source PV name>/<volume-snapshot-
content-name>
  volumeSnapshotRef:
    name: import-snap
    namespace: default
```

2. **Administrateur du cluster :** Créez la `VolumeSnapshot` CR qui référence l' `VolumeSnapshotContent`objet. Cela demande l'autorisation d'utiliser l' `VolumeSnapshot`dans un espace de noms donné.

   Exemple

   L'exemple suivant crée un `VolumeSnapshot` CR nommé `import-snap` qui référence le `VolumeSnapshotContent` nommé `import-snap-content`.

```
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshot
metadata:
  name: import-snap
spec:
  # volumeSnapshotClassName: csi-snapclass (not required for pre-
provisioned or imported snapshots)
  source:
    volumeSnapshotContentName: import-snap-content
```

3. **Traitement interne (aucune action requise) :** Le gestionnaire de snapshots externe reconnaît le nouvellement créé `VolumeSnapshotContent` et exécute l'appel `ListSnapshots`. Trident crée le `TridentSnapshot`.
   - Le dispositif de capture d'écran externe définit le `VolumeSnapshotContent` sur `readyToUse` et le `VolumeSnapshot` sur `true`.
   - Trident retourne `readyToUse=true`.
4. **Tout utilisateur :** Créez un `PersistentVolumeClaim` pour référencer le nouveau `VolumeSnapshot`, où le `spec.dataSource` (ou `spec.dataSourceRef`) nom est le `VolumeSnapshot` nom.

Exemple

L'exemple suivant crée un PVC faisant référence au `VolumeSnapshot` nommé `import-snap`.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-from-snap
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: simple-sc
  resources:
    requests:
      storage: 1Gi
  dataSource:
    name: import-snap
    kind: VolumeSnapshot
    apiGroup: snapshot.storage.k8s.io
```

Récupérer les données du volume à l'aide de snapshots

Le répertoire des instantanés est masqué par défaut afin d'assurer une compatibilité maximale des volumes provisionnés à l'aide des `ontap-nas` et `ontap-nas-economy` pilotes. Activez le répertoire `.snapshot` pour récupérer directement les données à partir des instantanés.

Utilisez la commande ONTAP volume snapshot restore pour restaurer un volume à un état enregistré dans un instantané précédent.

```
cluster1::*> volume snapshot restore -vserver vs0 -volume vol3 -snapshot
vol3_snap_archive
```

REMARQUE	Lorsque vous restaurez une copie instantanée, la configuration du volume existant est écrasée. Les modifications apportées aux données du volume après la création de la copie instantanée sont perdues.

Restauration sur place du volume à partir d'un instantané

Trident permet une restauration rapide et directe des volumes à partir d'un instantané grâce à la `TridentActionSnapshotRestore` (TASR) CR. Cette CR fonctionne comme une action Kubernetes impérative et n'est pas conservée après la fin de l'opération.

Trident prend en charge la restauration d'instantanés sur les `ontap-san`, `ontap-san-economy`, `ontap-nas`, `ontap-nas-flexgroup`, `azure-netapp-files`, `google-cloud-netapp-volumes` et `solidfire-san` pilotes.

Avant de commencer

Vous devez disposer d'une PVC reliée et d'un instantané de volume disponible.

- Vérifiez que le statut du PVC est bound.

```
kubectl get pvc
```

- Vérifiez que l'instantané du volume est prêt à être utilisé.

```
kubectl get vs
```

Étapes

1. Créez la TASR CR. Cet exemple crée une CR pour PVC `pvc1` et volume snapshot `pvc1-snapshot`.

   REMARQUE Le TASR CR doit se trouver dans un espace de noms où le PVC et le VS existent.

```
cat tasr-pvc1-snapshot.yaml
```

```
apiVersion: trident.netapp.io/v1
kind: TridentActionSnapshotRestore
metadata:
  name: trident-snap
  namespace: trident
spec:
  pvcName: pvc1
  volumeSnapshotName: pvc1-snapshot
```

2. Appliquez la CR pour restaurer à partir de l'instantané. Cet exemple restaure à partir de l'instantané `pvc1`.

```
kubectl create -f tasr-pvc1-snapshot.yaml
```

```
tridentactionsnapshotrestore.trident.netapp.io/trident-snap created
```

Résultats

Trident restaure les données à partir de l'instantané. Vous pouvez vérifier l'état de la restauration de l'instantané :

```
kubectl get tasr -o yaml
```

```
apiVersion: trident.netapp.io/v1
items:
- apiVersion: trident.netapp.io/v1
  kind: TridentActionSnapshotRestore
  metadata:
    creationTimestamp: "2023-04-14T00:20:33Z"
    generation: 3
    name: trident-snap
    namespace: trident
    resourceVersion: "3453847"
    uid: <uid>
  spec:
    pvcName: pvc1
    volumeSnapshotName: pvc1-snapshot
  status:
    startTime: "2023-04-14T00:20:34Z"
    completionTime: "2023-04-14T00:20:37Z"
    state: Succeeded
kind: List
metadata:
  resourceVersion: ""
```

REMARQUE

- Dans la plupart des cas, Trident ne relancera pas automatiquement l'opération en cas d'échec. Vous devrez effectuer l'opération à nouveau.
- Les utilisateurs de Kubernetes ne disposant pas d'un accès administrateur peuvent devoir obtenir l'autorisation de l'administrateur pour créer un TASR CR dans l'espace de noms de leur application.

Supprimez un PV avec les instantanés associés

Lors de la suppression d'un volume persistant avec des snapshots associés, le volume Trident correspondant passe à l'état « Suppression en cours ». Supprimez les snapshots du volume pour supprimer le volume Trident.

Déployer un contrôleur d'instantané de volume

Si votre distribution Kubernetes n'inclut pas le snapshot controller et les CRDs, vous pouvez les déployer comme suit.

Étapes

1. Créer des CRD d'instantané de volume.

```
cat snapshot-setup.sh
```

```
#!/bin/bash
# Create volume snapshot CRDs
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
6.1/client/config/crd/snapshot.storage.k8s.io_volumesnapshotclasses.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
6.1/client/config/crd/snapshot.storage.k8s.io_volumesnapshotcontents.yam
l
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
6.1/client/config/crd/snapshot.storage.k8s.io_volumesnapshots.yaml
```

2. Créez le contrôleur de snapshot.

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-6.1/deploy/kubernetes/snapshot-
controller/rbac-snapshot-controller.yaml
```

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-6.1/deploy/kubernetes/snapshot-
controller/setup-snapshot-controller.yaml
```

REMARQUE	Si nécessaire, ouvrez `deploy/kubernetes/snapshot-controller/rbac-snapshot-controller.yaml` et mettez à jour `namespace` pour votre espace de noms.

Liens associés

- "Instantanés de volume"
- "VolumeSnapshotClass"

Travailler avec les instantanés de groupe de volumes

Les instantanés de groupes de volumes Kubernetes des volumes persistants (PV) NetApp Trident offrent la possibilité de créer des instantanés de plusieurs volumes (un groupe d'instantanés de volumes). Cet instantané de groupe de volumes représente des copies de plusieurs volumes prises au même moment.

REMARQUE	VolumeGroupSnapshot est une fonctionnalité bêta de Kubernetes avec des API bêta. Kubernetes 1.32 est la version minimale requise pour VolumeGroupSnapshot.

Créer des instantanés de groupes de volumes

La prise en charge des instantanés de groupes de volumes est assurée avec les pilotes de stockage suivants :

- `ontap-san` driver - uniquement pour les protocoles iSCSI et FC, pas pour le protocole NVMe/TCP.
- `ontap-san-economy` - uniquement pour le protocole iSCSI.
- `ontap-nas`

REMARQUE	La création d'instantanés de groupes de volumes n'est pas prise en charge pour NetApp ASA r2 ou les systèmes de stockage AFX.

Avant de commencer

- Assurez-vous que votre version de Kubernetes est K8s 1.32 ou supérieure.
- Vous devez disposer d'un contrôleur de snapshots externe et de Custom Resource Definitions (CRDs) pour travailler avec les snapshots. C'est la responsabilité de l'orchestrateur Kubernetes (par exemple : Kubeadm, GKE, OpenShift).

  Si votre distribution Kubernetes n'inclut pas le contrôleur de snapshots externe et les CRD, reportez-vous à Déployer un contrôleur d'instantané de volume.

REMARQUE	Ne créez pas de contrôleur de snapshots si vous créez des snapshots de groupes de volumes à la demande dans un environnement GKE. GKE utilise un contrôleur de snapshots intégré et masqué.

- Dans le fichier YAML du contrôleur d'instantané, définissez la `CSIVolumeGroupSnapshot`feature gate sur « true » pour garantir que l'instantané du groupe de volumes est activé.
- Créez les classes d'instantané de groupe de volumes requises avant de créer un instantané de groupe de volumes.
- Assurez-vous que tous les PVC/volumes se trouvent sur le même SVM pour pouvoir créer VolumeGroupSnapshot.

Étapes

- Créez un VolumeGroupSnapshotClass avant de créer un VolumeGroupSnapshot. Pour plus d'informations, consultez "VolumeGroupSnapshotClass".

```
apiVersion: groupsnapshot.storage.k8s.io/v1beta1
kind: VolumeGroupSnapshotClass
metadata:
  name: csi-group-snap-class
  annotations:
    kubernetes.io/description: "Trident group snapshot class"
driver: csi.trident.netapp.io
deletionPolicy: Delete
```

- Créez des PVC avec les étiquettes requises en utilisant les classes de stockage existantes, ou ajoutez ces étiquettes aux PVC existants.

L'exemple suivant crée le PVC en utilisant `pvc1-group-snap` comme source de données et l'étiquette `consistentGroupSnapshot: groupA`. Définissez la clé et la valeur de l'étiquette en fonction de vos besoins.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc1-group-snap
  labels:
    consistentGroupSnapshot: groupA
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 100Mi
  storageClassName: sc1-1
```

- Créez un VolumeGroupSnapshot avec la même étiquette (`consistentGroupSnapshot: groupA`) spécifiée dans le PVC.

  Cet exemple crée un instantané de groupe de volumes :

```
apiVersion: groupsnapshot.storage.k8s.io/v1beta1
kind: VolumeGroupSnapshot
metadata:
  name: "vgs1"
  namespace: trident
spec:
  volumeGroupSnapshotClassName: csi-group-snap-class
  source:
    selector:
      matchLabels:
        consistentGroupSnapshot: groupA
```

Récupérer les données du volume à l'aide d'un instantané de groupe

Vous pouvez restaurer des volumes persistants individuels à l'aide des instantanés individuels qui ont été créés dans le cadre de l'instantané de groupe de volumes. Vous ne pouvez pas restaurer l'instantané de groupe de volumes en tant qu'unité.

Utilisez la commande ONTAP volume snapshot restore pour restaurer un volume à un état enregistré dans un instantané précédent.

```
cluster1::*> volume snapshot restore -vserver vs0 -volume vol3 -snapshot
vol3_snap_archive
```

REMARQUE	Lorsque vous restaurez une copie instantanée, la configuration du volume existant est écrasée. Les modifications apportées aux données du volume après la création de la copie instantanée sont perdues.

Restauration sur place du volume à partir d'un instantané

Trident permet une restauration rapide et directe des volumes à partir d'un instantané grâce à la `TridentActionSnapshotRestore` (TASR) CR. Cette CR fonctionne comme une action Kubernetes impérative et n'est pas conservée après la fin de l'opération.

Pour plus d'informations, consultez "Restauration sur place du volume à partir d'un instantané".

Supprimer un PV avec des snapshots de groupe associés

Lors de la suppression d'un instantané de volume de groupe :

- Vous pouvez supprimer VolumeGroupSnapshots dans leur ensemble, pas les instantanés individuels du groupe.
- Si des PersistentVolumes sont supprimés alors qu'un instantané existe pour ce PersistentVolume, Trident déplacera ce volume vers un état « en cours de suppression » car l'instantané doit être supprimé avant que le volume puisse être supprimé en toute sécurité.
- Si un clone a été créé à l'aide d'un instantané groupé et que le groupe doit ensuite être supprimé, une opération de division sur le clone commencera et le groupe ne pourra pas être supprimé tant que la division n'est pas terminée.

Déployer un contrôleur d'instantané de volume

Si votre distribution Kubernetes n'inclut pas le snapshot controller et les CRDs, vous pouvez les déployer comme suit.

Étapes

1. Créer des CRD d'instantané de volume.

```
cat snapshot-setup.sh
```

```
#!/bin/bash
# Create volume snapshot CRDs
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
8.2/client/config/crd/groupsnapshot.storage.k8s.io_volumegroupsnapshotcl
asses.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
8.2/client/config/crd/groupsnapshot.storage.k8s.io_volumegroupsnapshotco
ntents.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
8.2/client/config/crd/groupsnapshot.storage.k8s.io_volumegroupsnapshots.
yaml
```

2. Créez le contrôleur de snapshot.

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-8.2/deploy/kubernetes/snapshot-
controller/rbac-snapshot-controller.yaml
```

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-8.2/deploy/kubernetes/snapshot-
controller/setup-snapshot-controller.yaml
```

REMARQUE	Si nécessaire, ouvrez `deploy/kubernetes/snapshot-controller/rbac-snapshot-controller.yaml` et mettez à jour `namespace` pour votre espace de noms.

Liens associés

- "VolumeGroupSnapshotClass"
- "Instantanés de volume"