



Avec Astra Trident

Astra Trident

NetApp
April 16, 2024

Sommaire

Avec Astra Trident	1
Configuration des systèmes back-end	1
Création de systèmes back-end avec kubectl	75
Effectuer la gestion back-end avec kubectl	82
Gestion back-end avec tridentctl	83
Passez d'une option de gestion back-end à une autre	85
Gérer les classes de stockage	91
Réaliser des opérations de volume	93
Préparez le nœud de travail	118
Préparation automatique du nœud de travail	122
Contrôle d'Astra Trident	122

Avec Astra Trident

Configuration des systèmes back-end

Un système back-end définit la relation entre Astra Trident et un système de stockage. Il explique à Astra Trident comment communiquer avec ce système de stockage et comment Astra Trident doit provisionner des volumes à partir de celui-ci. Astra Trident proposera automatiquement des pools de stockage provenant des systèmes back-end qui correspondent aux exigences définies par une classe de stockage. En savoir plus sur la configuration du système back-end en fonction du type de système de stockage dont vous disposez.

- ["Configurer un back-end Azure NetApp Files"](#)
- ["Configurer un serveur Cloud Volumes Service pour le système back-end AWS"](#)
- ["Configurer un système back-end Cloud Volumes Service pour Google Cloud Platform"](#)
- ["Configurer un système NetApp HCI ou SolidFire backend"](#)
- ["Configurer un système back-end avec des pilotes NAS ONTAP ou Cloud Volumes ONTAP"](#)
- ["Configurer un système back-end avec des pilotes ONTAP ou Cloud Volumes ONTAP SAN"](#)
- ["Utilisez Astra Trident avec Amazon FSX pour NetApp ONTAP"](#)

Configurer un back-end Azure NetApp Files

Découvrez comment configurer Azure NetApp Files (ANF) en tant que backend pour votre installation d'Astra Trident à l'aide des exemples de configuration fournis.



Le service Azure NetApp Files ne prend pas en charge des volumes inférieurs à 100 Go. Astra Trident crée automatiquement des volumes de 100 Go en cas de demande d'un volume plus petit.

Ce dont vous avez besoin

Pour configurer et utiliser un ["Azure NetApp Files"](#) back-end, vous avez besoin des éléments suivants :

- `subscriptionID` Depuis un abonnement Azure avec Azure NetApp Files activé.
- `tenantID`, `clientID`, et `clientSecret` à partir d'un ["Enregistrement d'applications"](#) Dans Azure Active Directory avec les autorisations suffisantes pour le service Azure NetApp Files. L'enregistrement de l'application doit utiliser le `Owner` ou `Contributor` Rôle prédéfini par Azure.



Pour en savoir plus sur les rôles intégrés dans Azure, consultez la ["Documentation Azure"](#).

- Azure `location` qui contient au moins un ["sous-réseau délégué"](#).
- Si vous utilisez Azure NetApp Files pour la première fois ou à un nouvel emplacement, une certaine configuration initiale est requise. Voir la ["guide de démarrage rapide"](#).

Description de la tâche

En fonction de la configuration back-end (sous-réseau, réseau virtuel, niveau de service et emplacement), Trident crée des volumes ANF dans des pools de capacité disponibles à l'emplacement demandé et qui correspondent au niveau de service et au sous-réseau requis.



Avec Astra Trident 21.04.0 et les versions antérieures, il ne prend pas en charge les pools de capacité manuels de QoS.

Options de configuration du back-end

Voir le tableau suivant pour les options de configuration du back-end :

Paramètre	Description	Valeur par défaut
version		Toujours 1
storageDriverName	Nom du pilote de stockage	« azure-netapp-files »
backendName	Nom personnalisé ou système back-end de stockage	Nom du pilote + "_" + caractères aléatoires
subscriptionID	L'ID d'abonnement de votre abonnement Azure	
tenantID	ID locataire d'un enregistrement d'application	
clientID	L'ID client d'un enregistrement d'application	
clientSecret	Secret client d'un enregistrement d'application	
serviceLevel	Un de Standard, Premium, ou Ultra	« » (aléatoire)
location	Nom de l'emplacement Azure dans lequel les nouveaux volumes seront créés	« » (aléatoire)
virtualNetwork	Nom d'un réseau virtuel avec un sous-réseau délégué	« » (aléatoire)
subnet	Nom d'un sous-réseau délégué à Microsoft.Netapp/volumes	« » (aléatoire)
nfsMountOptions	Contrôle précis des options de montage NFS.	« nfsvers=3 »
limitVolumeSize	Echec du provisionnement si la taille du volume demandé est supérieure à cette valeur	« » (non appliqué par défaut)
debugTraceFlags	Indicateurs de débogage à utiliser lors du dépannage. Exemple <code>\{"api":false, "method":true}</code> . Ne l'utilisez pas à moins que vous ne soyez en mesure de résoudre les problèmes et que vous ayez besoin d'un vidage détaillé des journaux.	nul



Modification du `capacityPools` de manière à réduire le nombre de pools de capacité utilisés pour le provisionnement, les volumes orphelins qui sont provisionnés sur le pool/pools de capacité qui ne font pas partie du `capacityPools` liste désormais disponible. Les opérations de clonage de ces volumes orphelins échouent.



Si vous rencontrez une erreur "aucun pool de capacité détecté" lors de la tentative de création d'une demande de volume persistant, il est probable que votre enregistrement d'application ne dispose pas des autorisations et ressources requises (sous-réseau, réseau virtuel, pool de capacité) associées. Astra Trident consigne les ressources Azure qu'il a découvertes lors de la création du système back-end lorsque le débogage est activé. Assurez-vous de vérifier si un rôle approprié est utilisé.



Si vous souhaitez monter des volumes à l'aide de la version 4.1 de NFS, vous pouvez inclure `nfsvers=4` Dans la liste des options de montage délimitées par des virgules, choisissez NFS v4.1. Toutes les options de montage définies dans une classe de stockage remplacent les options de montage définies dans un fichier de configuration backend.

Vous pouvez contrôler la manière dont chaque volume est provisionné par défaut en spécifiant les options suivantes dans une section spéciale du fichier de configuration. Voir les exemples de configuration ci-dessous.

Paramètre	Description	Valeur par défaut
<code>exportRule</code>	Règle(s) d'exportation pour les nouveaux volumes	« 0.0.0.0/0 »
<code>size</code>	Taille par défaut des nouveaux volumes	« 100 G »

Le `exportRule` La valeur doit être une liste séparée par des virgules d'une combinaison d'adresses IPv4 ou de sous-réseaux IPv4 en notation CIDR.



Pour tous les volumes créés sur un back-end ANF, Astra Trident copie tous les libellés présents sur un pool de stockage vers le volume de stockage au moment du provisionnement. Les administrateurs de stockage peuvent définir des étiquettes par pool de stockage et regrouper tous les volumes créés dans un pool de stockage. Cela permet de différencier facilement les volumes en fonction d'un ensemble d'étiquettes personnalisables fournies dans la configuration back-end.

Exemple 1 : configuration minimale

Il s'agit de la configuration back-end minimale absolue. Avec cette configuration, Astra Trident détecte tous vos comptes, pools de capacité et sous-réseaux NetApp délégués à ANF dans le monde entier, et place de nouveaux volumes de façon aléatoire.

Cette configuration est idéale pour commencer avec ANF et essayer certaines choses. Toutefois, dans la pratique, vous voulez fournir des fonctionnalités supplémentaires pour déterminer les volumes que vous provisionnez.

```
{
  "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"
}
```

Exemple 2 : configuration à un seul emplacement et niveau de service spécifique

Cette configuration back-end place les volumes dans des Azure eastus emplacement dans un Premium pool de capacité. Astra Trident détecte automatiquement tous les sous-réseaux délégués à ANF dans cet emplacement et place un nouveau volume de façon aléatoire sur l'un d'entre eux.

```
{
  "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": "Premium"
}
```

Exemple 3 : configuration avancée

Cette configuration back-end réduit davantage l'étendue du placement des volumes sur un seul sous-réseau et modifie également certains paramètres par défaut du 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": "Premium",
  "virtualNetwork": "my-virtual-network",
  "subnet": "my-subnet",
  "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",
    "size": "200Gi"
  }
}
```

Exemple 4 : configuration de pool de stockage virtuel

Cette configuration back-end définit plusieurs pools de stockage dans un seul fichier. Cette fonction est utile lorsque plusieurs pools de capacité prennent en charge différents niveaux de service, et que vous souhaitez créer des classes de stockage dans Kubernetes qui les représentent.

```

{
  "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",
  "nfsMountOptions": "vers=3,proto=tcp,timeo=600",
  "labels": {
    "cloud": "azure"
  },
  "location": "eastus",

  "storage": [
    {
      "labels": {
        "performance": "gold"
      },
      "serviceLevel": "Ultra"
    },
    {
      "labels": {
        "performance": "silver"
      },
      "serviceLevel": "Premium"
    },
    {
      "labels": {
        "performance": "bronze"
      },
      "serviceLevel": "Standard",
    }
  ]
}

```

Les éléments suivants `StorageClass` les définitions font référence aux pools de stockage ci-dessus. À l'aide du `parameters.selector` vous pouvez spécifier pour chaque champ `StorageClass` pool virtuel utilisé pour héberger un volume. Les aspects définis dans le pool sélectionné seront définis pour le volume.


```
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
```

Et la suite ?

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

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

Si la création du back-end échoue, la configuration du back-end est erronée. Vous pouvez afficher les journaux pour déterminer la cause en exécutant la commande suivante :

```
tridentctl logs
```

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

Configurer CVS pour le back-end AWS

Découvrez comment configurer NetApp Cloud Volumes Service (CVS) pour AWS en tant que backend avec

vosre installation d'Astra Trident à l'aide des exemples de configuration fournis.



Cloud Volumes Service pour AWS ne prend pas en charge les volumes inférieurs à 100 Go. Trident crée automatiquement des volumes de 100 Go lorsqu'un volume plus petit est demandé.

Ce dont vous avez besoin

Pour configurer et utiliser le "Cloud Volumes Service pour AWS" back-end, vous avez besoin des éléments suivants :

- Un compte AWS configuré avec NetApp CVS
- Région API, URL et clés pour votre compte CVS

Options de configuration du back-end

Voir le tableau suivant pour les options de configuration du back-end :

Paramètre	Description	Valeur par défaut
version		Toujours 1
storageDriverName	Nom du pilote de stockage	« aws-cvs »
backendName	Nom personnalisé ou système back-end de stockage	Nom du pilote + "_" + partie de la clé API
apiRegion	Région de compte CVS. Vous pouvez trouver la valeur dans le portail Web CVS dans Paramètres de compte/accès API.	
apiURL	URL DE L'API DU COMPTE CVS. Vous pouvez trouver la valeur dans le portail Web CVS dans Paramètres de compte/accès API.	
apiKey	Clé API de compte CVS. Vous pouvez trouver la valeur dans le portail Web CVS dans Paramètres de compte/accès API.	
secretKey	Clé secrète compte CVS. Vous pouvez trouver la valeur dans le portail Web CVS dans Paramètres de compte/accès API.	
proxyURL	URL proxy si le serveur proxy doit se connecter au compte CVS. Le serveur proxy peut être un proxy HTTP ou HTTPS. Pour un proxy HTTPS, la validation du certificat est ignorée pour permettre l'utilisation de certificats auto-signés dans le serveur proxy. Les serveurs proxy avec authentification activée ne sont pas pris en charge.	

Paramètre	Description	Valeur par défaut
nfsMountOptions	Contrôle précis des options de montage NFS.	« nfsvers=3 »
limitVolumeSize	Echec du provisionnement si la taille du volume demandé est supérieure à cette valeur	« » (non appliqué par défaut)
serviceLevel	Niveau de service CVS pour les nouveaux volumes. Les valeurs sont « standard », « Premium » et « extrême ».	« standard »
debugTraceFlags	Indicateurs de débogage à utiliser lors du dépannage. Exemple <code>\{"api":false, "method":true}</code> . Ne l'utilisez pas à moins que vous ne soyez en mesure de résoudre les problèmes et que vous ayez besoin d'un vidage détaillé des journaux.	nul



apiURL est unique pour chaque apiRegion. Par exemple, le US-West-2 apiRegion a le <https://cv.us-west-2.netapp.com:8080/v1/> apiURL. De même, le US-est-1 apiRegion a le <https://cde-aws-bundles.netapp.com:8080/v1/> apiURL. Assurez-vous de vérifier que le tableau de bord CVS est correct apiRegion et apiURL paramètres de votre configuration backend.

Chaque système back-end provisionne les volumes dans une seule région AWS. Pour créer des volumes dans d'autres régions, vous pouvez définir des systèmes back-end supplémentaires.

Vous pouvez contrôler la manière dont chaque volume est provisionné par défaut en spécifiant les options suivantes dans une section spéciale du fichier de configuration. Voir les exemples de configuration ci-dessous.

Paramètre	Description	Valeur par défaut
exportRule	Règle(s) d'exportation pour les nouveaux volumes	« 0.0.0.0/0 »
snapshotDir	Contrôle la visibilité du .snapshot répertoire	« faux »
snapshotReserve	Pourcentage de volume réservé pour les snapshots	« » (Accepter CVS par défaut de 0)
size	La taille des nouveaux volumes	« 100 G »

Le exportRule La valeur doit être une liste séparée par des virgules d'une combinaison d'adresses IPv4 ou de sous-réseaux IPv4 en notation CIDR.



Pour tous les volumes créés sur un back-end CVS AWS, Astra Trident copie toutes les étiquettes présentes sur un pool de stockage vers le volume de stockage au moment du provisionnement. Les administrateurs de stockage peuvent définir des étiquettes par pool de stockage et regrouper tous les volumes créés dans un pool de stockage. Cela permet de différencier facilement les volumes en fonction d'un ensemble d'étiquettes personnalisables fournies dans la configuration back-end.

Exemple 1 : configuration minimale

Il s'agit de la configuration back-end minimale absolue.

Cette configuration est idéale lorsque vous commencez avec CVS AWS et que vous essayez les choses. Toutefois, en pratique, vous devez fournir une évaluation supplémentaire des volumes que vous provisionnez.

```
{
  "version": 1,
  "storageDriverName": "aws-cvs",
  "apiRegion": "us-east-1",
  "apiURL": "https://cds-aws-bundles.netapp.com:8080/v1",
  "apiKey": "znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGz1zZE",
  "secretKey": "rR0rUmWXfNioN1KhtHisIsAnoTherboGuskey6pU"
}
```

Exemple 2 : configuration de niveau de service unique

Dans cet exemple, nous avons présenté un fichier back-end qui applique les mêmes aspects à tous les systèmes de stockage créés par Astra Trident dans la région AWS US-East-1. Cet exemple montre également l'utilisation de `proxyURL` dans le fichier backend.

```

{
  "version": 1,
  "storageDriverName": "aws-cvs",
  "backendName": "cvs-aws-us-east",
  "apiRegion": "us-east-1",
  "apiURL": "https://cvs-aws-bundles.netapp.com:8080/v1",
  "apiKey": "znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGz1zZE",
  "secretKey": "rR0rUmWXfNioN1KhtHisIsAnoTherboGuskey6pU",
  "proxyURL": "http://proxy-server-hostname/",
  "nfsMountOptions": "vers=3,proto=tcp,timeo=600",
  "limitVolumeSize": "50Gi",
  "serviceLevel": "premium",
  "defaults": {
    "snapshotDir": "true",
    "snapshotReserve": "5",
    "exportRule": "10.0.0.0/24,10.0.1.0/24,10.0.2.100",
    "size": "200Gi"
  }
}

```

Exemple 3 : configuration de pool de stockage virtuel

Cet exemple représente le fichier de définition back-end configuré avec des pools de stockage virtuel et des classes de stockage qui les renvoient.

Dans l'exemple de fichier de définition de back-end illustré ci-dessous, des valeurs par défaut spécifiques sont définies pour tous les pools de stockage, qui définissent le `snapshotReserve` à 5 % et le `exportRule` à 0.0.0.0/0. Les pools de stockage virtuels sont définis dans le `storage` section. Dans cet exemple, chaque pool de stockage est défini lui-même `serviceLevel`, et certains pools remplacent les valeurs par défaut.

```

{
  "version": 1,
  "storageDriverName": "aws-cvs",
  "apiRegion": "us-east-1",
  "apiURL": "https://cvs-aws-bundles.netapp.com:8080/v1",
  "apiKey": "EnterYourAPIKeyHere*****",
  "secretKey": "EnterYourSecretKeyHere*****",
  "nfsMountOptions": "vers=3,proto=tcp,timeo=600",

  "defaults": {
    "snapshotReserve": "5",
    "exportRule": "0.0.0.0/0"
  },

  "labels": {
    "cloud": "aws"
  }
}

```

```

},
"region": "us-east-1",

"storage": [
  {
    "labels": {
      "performance": "extreme",
      "protection": "extra"
    },
    "serviceLevel": "extreme",
    "defaults": {
      "snapshotDir": "true",
      "snapshotReserve": "10",
      "exportRule": "10.0.0.0/24"
    }
  },
  {
    "labels": {
      "performance": "extreme",
      "protection": "standard"
    },
    "serviceLevel": "extreme"
  },
  {
    "labels": {
      "performance": "premium",
      "protection": "extra"
    },
    "serviceLevel": "premium",
    "defaults": {
      "snapshotDir": "true",
      "snapshotReserve": "10"
    }
  },
  {
    "labels": {
      "performance": "premium",
      "protection": "standard"
    },
    "serviceLevel": "premium"
  },
  {
    "labels": {
      "performance": "standard"
    }
  }
]

```

```

    },
    "serviceLevel": "standard"
  }
]
}

```

Les définitions de classe de stockage suivantes font référence aux pools de stockage ci-dessus. À l'aide du `parameters.selector` Vous pouvez spécifier pour chaque classe de stockage le pool virtuel utilisé pour héberger un volume. Les aspects définis dans le pool sélectionné seront définis pour le volume.

La première classe de stockage (`cvs-extreme-extra-protection`) correspond au premier pool de stockage virtuel. Il s'agit du seul pool offrant des performances extrêmes avec une réserve Snapshot de 10 %. La dernière classe de stockage (`cvs-extra-protection`) appelle tout pool de stockage qui fournit une réserve d'instantanés de 10%. Astra Trident décide du pool de stockage virtuel sélectionné et s'assure que les exigences de la réserve Snapshot sont respectées.

```

apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: cvs-extreme-extra-protection
provisioner: netapp.io/trident
parameters:
  selector: "performance=extreme; protection=extra"
allowVolumeExpansion: true
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: cvs-extreme-standard-protection
provisioner: netapp.io/trident
parameters:
  selector: "performance=extreme; protection=standard"
allowVolumeExpansion: true
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: cvs-premium-extra-protection
provisioner: netapp.io/trident
parameters:
  selector: "performance=premium; protection=extra"
allowVolumeExpansion: true
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:

```

```
name: cvs-premium
provisioner: netapp.io/trident
parameters:
  selector: "performance=premium; protection=standard"
allowVolumeExpansion: true
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: cvs-standard
provisioner: netapp.io/trident
parameters:
  selector: "performance=standard"
allowVolumeExpansion: true
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: cvs-extra-protection
provisioner: netapp.io/trident
parameters:
  selector: "protection=extra"
allowVolumeExpansion: true
```

Et la suite ?

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

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

Si la création du back-end échoue, la configuration du back-end est erronée. Vous pouvez afficher les journaux pour déterminer la cause en exécutant la commande suivante :

```
tridentctl logs
```

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

Configurer CVS pour le back-end GCP

Découvrez comment configurer NetApp Cloud Volumes Service (CVS) pour Google Cloud Platform (GCP) en tant que backend avec votre installation d'Astra Trident, à l'aide des exemples de configurations fournis.



NetApp Cloud Volumes Service pour Google Cloud ne prend pas en charge les volumes CVS-Performance de moins de 100 Gio, ou les volumes CVS de moins de 300 Gio. Astra Trident crée automatiquement des volumes de taille minimale si le volume demandé est inférieur à la taille minimale.

Ce dont vous avez besoin

Pour configurer et utiliser le "Cloud Volumes Service pour Google Cloud" back-end, vous avez besoin des éléments suivants :

- Un compte Google Cloud configuré avec NetApp CVS
- Numéro de projet de votre compte Google Cloud
- Compte de service Google Cloud avec le `netappcloudvolumes.admin` rôle
- Fichier de clé API pour votre compte de service CVS

Astra Trident inclut désormais la prise en charge par défaut des volumes plus petits "Type de service CVS sur GCP". Pour les systèmes back-end créés avec `storageClass=software`, Les volumes auront une taille de provisionnement minimale de 300 Gio. CVS offre actuellement cette fonctionnalité sous disponibilité contrôlée et ne fournit pas de support technique. Les utilisateurs doivent s'inscrire pour accéder aux volumes de sous-Tio "ici". NetApp recommande aux clients d'utiliser des volumes de sous-Tio pour **charges de travail non-production**.



Lors du déploiement des systèmes back-end avec le type de service CVS par défaut (`storageClass=software`), les utilisateurs doivent obtenir l'accès à la fonctionnalité de volumes de sous-Tio dans GCP pour le(s) numéro(s) de projet et l'ID de projet en question. Il est nécessaire qu'Astra Trident provisionne des volumes de sous-Tio. Si ce n'est pas le cas, les créations de volume échoueront pour les demandes de volume dont la taille est inférieure à 600 Gio. Obtenir l'accès aux volumes de sous-Tio à l'aide de "ce formulaire".

Les volumes créés par Astra Trident pour le niveau de service CVS par défaut seront provisionnés comme suit :

- Si la quantité de PVC est inférieure à 300 Gio, l'Astra Trident crée un volume CVS de 300 Gio.
- Avec des demandes de volume persistant comprises entre 300 Gio et 600 Gio, l'Astra Trident crée un volume CVS de la taille demandée.
- Si les demandes de volume persistant sont comprises entre 600 Gio et 1 Tio, l'Astra Trident crée un volume CVS Tio.
- Avec des demandes de volume supérieur à 1 Tio, Astra Trident crée un volume CVS de la taille demandée.

Options de configuration du back-end

Voir le tableau suivant pour les options de configuration du back-end :

Paramètre	Description	Valeur par défaut
<code>version</code>		Toujours 1
<code>storageDriverName</code>	Nom du pilote de stockage	« gcp-cvs »
<code>backendName</code>	Nom personnalisé ou système back-end de stockage	Nom du pilote + "_" + partie de la clé API

Paramètre	Description	Valeur par défaut
storageClass	Type de stockage. Choisissez parmi <code>hardware</code> (performances optimisées) ou <code>software</code> (Type de service CVS)	
projectNumber	Numéro de projet de compte Google Cloud. La valeur ajoutée est disponible sur la page d'accueil du portail Google Cloud.	
apiRegion	Région de compte CVS. Il s'agit de la région où le back-end provisionne les volumes.	
apiKey	Clé API pour le compte de service Google Cloud avec le <code>netappcloudvolumes.admin</code> rôle. Il inclut le contenu au format JSON du fichier de clé privée d'un compte de service Google Cloud (copié en compte dans le fichier de configuration back-end).	
proxyURL	URL proxy si le serveur proxy doit se connecter au compte CVS. Le serveur proxy peut être un proxy HTTP ou HTTPS. Pour un proxy HTTPS, la validation du certificat est ignorée pour permettre l'utilisation de certificats auto-signés dans le serveur proxy. Les serveurs proxy avec authentification activée ne sont pas pris en charge.	
nfsMountOptions	Contrôle précis des options de montage NFS.	« <code>nfsvers=3</code> »
limitVolumeSize	Echec du provisionnement si la taille du volume demandé est supérieure à cette valeur	« » (non appliqué par défaut)
serviceLevel	Niveau de service CVS pour les nouveaux volumes. Les valeurs sont « <code>standard</code> », « <code>Premium</code> » et « <code>extrême</code> ».	« <code>standard</code> »
network	Réseau GCP utilisé pour les volumes CVS	« <code>par défaut</code> »

Paramètre	Description	Valeur par défaut
debugTraceFlags	Indicateurs de débogage à utiliser lors du dépannage. Exemple <pre>\{"api":false, "method":true}</pre> . Ne l'utilisez pas à moins que vous ne soyez en mesure de résoudre les problèmes et que vous ayez besoin d'un vidage détaillé des journaux.	nul

Si vous utilisez un réseau VPC partagé, les deux `projectNumber` et `hostProjectNumber` doit être spécifié. Dans ce cas, `projectNumber` est le projet de service, et `hostProjectNumber` est le projet hôte.

Le `apiRegion` Il représente la région GCP dans laquelle Astra Trident crée des volumes CVS. Astra Trident peut monter et relier des volumes sur des nœuds Kubernetes qui appartiennent à la même région GCP. Lors de la création d'un back-end, il est important de s'assurer `apiRegion` Correspondance avec la région dans laquelle les nœuds Kubernetes sont déployés. Lors de la création de clusters Kubernetes multi-région, des volumes CVS sont créés dans un cluster donné `apiRegion` Elle n'est possible qu'avec des charges de travail planifiées sur des nœuds se trouvant dans la même région GCP.



`storageClass` est un paramètre facultatif que vous pouvez utiliser pour sélectionner le paramètre souhaité "Type de service CVS". Vous pouvez choisir parmi le type de service CVS de base (`storageClass=software`) Ou le type de service CVS-Performance (`storageClass=hardware`), que Trident utilise par défaut. Assurez-vous de spécifier un `apiRegion` Qui fournit le CVS respectif `storageClass` dans votre définition de back-end.



L'intégration d'Astra Trident avec le type de service CVS de base sur Google Cloud est une fonctionnalité **bêta**, non destinée aux workloads de production. Trident est **entièrement pris en charge** avec le type de service CVS-Performance et l'utilise par défaut.

Chaque back-end provisionne les volumes dans une seule région Google Cloud. Pour créer des volumes dans d'autres régions, vous pouvez définir des systèmes back-end supplémentaires.

Vous pouvez contrôler la manière dont chaque volume est provisionné par défaut en spécifiant les options suivantes dans une section spéciale du fichier de configuration. Voir les exemples de configuration ci-dessous.

Paramètre	Description	Valeur par défaut
<code>exportRule</code>	Règle(s) d'exportation pour les nouveaux volumes	« 0.0.0.0/0 »
<code>snapshotDir</code>	Accès au <code>.snapshot</code> répertoire	« faux »
<code>snapshotReserve</code>	Pourcentage de volume réservé pour les snapshots	« » (Accepter CVS par défaut de 0)
<code>size</code>	La taille des nouveaux volumes	« 100Gi »

Le `exportRule` La valeur doit être une liste séparée par des virgules d'une combinaison d'adresses IPv4 ou de sous-réseaux IPv4 en notation CIDR.


```

guSaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrtrHisIsAbOguSaPIKe
yAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrtrHisIsAbOguSaPIKeyAZNchRA
Gz1zZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrtrHisIsAbOguSaPIKeyAZNchRAGz1zZE4j
K3bl/qp8B4Kws8zX5ojY9m\nXsYg6gyxy4zq7OlwWgLwGa==\n-----END PRIVATE
KEY-----\n",
    "client_email": "cloudvolumes-admin-sa@my-gcp-
project.iam.gserviceaccount.com",
    "client_id": "123456789012345678901",
    "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/cloudvolumes-admin-
sa%40my-gcp-project.iam.gserviceaccount.com"
  }
}

```

Exemple 3 : configuration de niveau de service unique

Cet exemple montre un fichier back-end qui applique les mêmes aspects à tous les systèmes de stockage créés par Astra Trident dans la région Google Cloud US-west2. Cet exemple montre également l'utilisation de proxyURL dans le fichier de configuration back-end.

```

{
  "version": 1,
  "storageDriverName": "gcp-cvs",
  "projectNumber": "012345678901",
  "apiRegion": "us-west2",
  "apiKey": {
    "type": "service_account",
    "project_id": "my-gcp-project",
    "private_key_id": "1234567890123456789012345678901234567890",
    "private_key": "-----BEGIN PRIVATE KEY-----
\nznHczZsrtrHisIsAbOguSaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZ
srtrHisIsAbOguSaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrtrHisI
sAbOguSaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrtrHisIsAbOguSa
PIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrtrHisIsAbOguSaPIKeyAZN
chRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrtrHisIsAbOguSaPIKeyAZNchRAGz1z
ZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrtrHisIsAbOguSaPIKeyAZNchRAGz1zZE4jK3bl
/qp8B4Kws8zX5ojY9m\nznHczZsrtrHisIsAbOguSaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kw
s8zX5ojY9m\nznHczZsrtrHisIsAbOguSaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY
9m\nznHczZsrtrHisIsAbOguSaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHc
zZsrtrHisIsAbOguSaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrtrHi
sIsAbOguSaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrtrHisIsAbOgu
SaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrtrHisIsAbOguSaPIKeyA

```

```

ZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrtrHisIsAbOguSaPIKeyAZNchRAGz
1zZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrtrHisIsAbOguSaPIKeyAZNchRAGz1zZE4jK3
bl/qp8B4Kws8zX5ojY9m\nznHczZsrtrHisIsAbOguSaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4
Kws8zX5ojY9m\nznHczZsrtrHisIsAbOguSaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5o
jY9m\nznHczZsrtrHisIsAbOguSaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m\nzn
HczZsrtrHisIsAbOguSaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrtr
HisIsAbOguSaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrtrHisIsAbO
guSaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrtrHisIsAbOguSaPIKe
yAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrtrHisIsAbOguSaPIKeyAZNchRA
Gz1zZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrtrHisIsAbOguSaPIKeyAZNchRAGz1zZE4j
K3bl/qp8B4Kws8zX5ojY9m\nXsYg6gyxy4zq70lwWgLwGa==\n-----END PRIVATE
KEY-----\n",
    "client_email": "cloudvolumes-admin-sa@my-gcp-
project.iam.gserviceaccount.com",
    "client_id": "123456789012345678901",
    "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/cloudvolumes-admin-
sa%40my-gcp-project.iam.gserviceaccount.com"
  },
  "proxyURL": "http://proxy-server-hostname/",
  "nfsMountOptions": "vers=3,proto=tcp,timeo=600",
  "limitVolumeSize": "10Ti",
  "serviceLevel": "premium",
  "defaults": {
    "snapshotDir": "true",
    "snapshotReserve": "5",
    "exportRule": "10.0.0.0/24,10.0.1.0/24,10.0.2.100",
    "size": "5Ti"
  }
}

```

Exemple 4 : configuration de pool de stockage virtuel

Cet exemple représente le fichier de définition back-end configuré avec des pools de stockage virtuel et avec StorageClasses cela leur renvoie.

Dans l'exemple de fichier de définition de back-end illustré ci-dessous, des valeurs par défaut spécifiques sont définies pour tous les pools de stockage, qui définissent le `snapshotReserve` à 5 % et le `exportRule` à 0.0.0.0/0. Les pools de stockage virtuels sont définis dans le `storage` section. Dans cet exemple, chaque pool de stockage est défini lui-même `serviceLevel`, et certains pools remplacent les valeurs par défaut.

```
{
```

```

"version": 1,
"storageDriverName": "gcp-cvs",
"projectNumber": "012345678901",
"apiRegion": "us-west2",
"apiKey": {
  "type": "service_account",
  "project_id": "my-gcp-project",
  "private_key_id": "1234567890123456789012345678901234567890",
  "private_key": "-----BEGIN PRIVATE KEY-----
\nznHczZsrtrHisIsAbOguSaPIKeyAZNchRAGzllzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZ
srtrHisIsAbOguSaPIKeyAZNchRAGzllzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrtrHisI
sAbOguSaPIKeyAZNchRAGzllzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrtrHisIsAbOguSa
PIKeyAZNchRAGzllzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrtrHisIsAbOguSaPIKeyAZN
chRAGzllzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrtrHisIsAbOguSaPIKeyAZNchRAGzll
ZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrtrHisIsAbOguSaPIKeyAZNchRAGzllzZE4jK3bl
/qp8B4Kws8zX5ojY9m\nznHczZsrtrHisIsAbOguSaPIKeyAZNchRAGzllzZE4jK3bl/qp8B4Kw
s8zX5ojY9m\nznHczZsrtrHisIsAbOguSaPIKeyAZNchRAGzllzZE4jK3bl/qp8B4Kws8zX5ojY
9m\nznHczZsrtrHisIsAbOguSaPIKeyAZNchRAGzllzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHc
zZsrtrHisIsAbOguSaPIKeyAZNchRAGzllzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrtrH
isIsAbOguSaPIKeyAZNchRAGzllzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrtrHisIsAbOgu
SaPIKeyAZNchRAGzllzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrtrHisIsAbOguSaPIKeyA
ZNchRAGzllzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrtrHisIsAbOguSaPIKeyAZNchRAGz
llzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrtrHisIsAbOguSaPIKeyAZNchRAGzllzZE4jK3
bl/qp8B4Kws8zX5ojY9m\nznHczZsrtrHisIsAbOguSaPIKeyAZNchRAGzllzZE4jK3bl/qp8B4
Kws8zX5ojY9m\nznHczZsrtrHisIsAbOguSaPIKeyAZNchRAGzllzZE4jK3bl/qp8B4Kws8zX5o
jY9m\nznHczZsrtrHisIsAbOguSaPIKeyAZNchRAGzllzZE4jK3bl/qp8B4Kws8zX5ojY9m\nzn
HczZsrtrHisIsAbOguSaPIKeyAZNchRAGzllzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrtr
HisIsAbOguSaPIKeyAZNchRAGzllzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrtrHisIsAbO
guSaPIKeyAZNchRAGzllzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrtrHisIsAbOguSaPIKe
yAZNchRAGzllzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrtrHisIsAbOguSaPIKeyAZNchRA
GzllzZE4jK3bl/qp8B4Kws8zX5ojY9m\nznHczZsrtrHisIsAbOguSaPIKeyAZNchRAGzllzZE4j
K3bl/qp8B4Kws8zX5ojY9m\nXsYg6gyxy4zq70lwWgLwGa==\n-----END PRIVATE
KEY-----\n",
  "client_email": "cloudvolumes-admin-sa@my-gcp-
project.iam.gserviceaccount.com",
  "client_id": "123456789012345678901",
  "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/cloudvolumes-admin-
sa%40my-gcp-project.iam.gserviceaccount.com"
},
"nfsMountOptions": "vers=3,proto=tcp,timeo=600",

```



```

"defaults": {
  "snapshotReserve": "5",
  "exportRule": "0.0.0.0/0"
},

"labels": {
  "cloud": "gcp"
},
"region": "us-west2",

"storage": [
  {
    "labels": {
      "performance": "extreme",
      "protection": "extra"
    },
    "serviceLevel": "extreme",
    "defaults": {
      "snapshotDir": "true",
      "snapshotReserve": "10",
      "exportRule": "10.0.0.0/24"
    }
  },
  {
    "labels": {
      "performance": "extreme",
      "protection": "standard"
    },
    "serviceLevel": "extreme"
  },
  {
    "labels": {
      "performance": "premium",
      "protection": "extra"
    },
    "serviceLevel": "premium",
    "defaults": {
      "snapshotDir": "true",
      "snapshotReserve": "10"
    }
  },
  {
    "labels": {
      "performance": "premium",
      "protection": "standard"
    }
  }
]

```

```

    },
    "serviceLevel": "premium"
  },
  {
    "labels": {
      "performance": "standard"
    },
    "serviceLevel": "standard"
  }
]
}

```

Les définitions de classe de stockage suivantes font référence aux pools de stockage ci-dessus. À l'aide du `parameters.selector` Vous pouvez spécifier pour chaque classe de stockage le pool virtuel utilisé pour héberger un volume. Les aspects définis dans le pool sélectionné seront définis pour le volume.

La première classe de stockage (`cvs-extreme-extra-protection`) correspond au premier pool de stockage virtuel. Il s'agit du seul pool offrant des performances extrêmes avec une réserve Snapshot de 10 %. La dernière classe de stockage (`cvs-extra-protection`) appelle tout pool de stockage qui fournit une réserve d'instantanés de 10%. Astra Trident décide du pool de stockage virtuel sélectionné et s'assure que les exigences de la réserve Snapshot sont respectées.

```

apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: cvs-extreme-extra-protection
provisioner: netapp.io/trident
parameters:
  selector: "performance=extreme; protection=extra"
allowVolumeExpansion: true
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: cvs-extreme-standard-protection
provisioner: netapp.io/trident
parameters:
  selector: "performance=premium; protection=standard"
allowVolumeExpansion: true
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: cvs-premium-extra-protection
provisioner: netapp.io/trident

```

```
parameters:
  selector: "performance=premium; protection=extra"
allowVolumeExpansion: true
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: cvs-premium
provisioner: netapp.io/trident
parameters:
  selector: "performance=premium; protection=standard"
allowVolumeExpansion: true
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: cvs-standard
provisioner: netapp.io/trident
parameters:
  selector: "performance=standard"
allowVolumeExpansion: true
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: cvs-extra-protection
provisioner: netapp.io/trident
parameters:
  selector: "protection=extra"
allowVolumeExpansion: true
```

Et la suite ?

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

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

Si la création du back-end échoue, la configuration du back-end est erronée. Vous pouvez afficher les journaux pour déterminer la cause en exécutant la commande suivante :

```
tridentctl logs
```

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

Configurer un système NetApp HCI ou SolidFire backend

Découvrez comment créer et utiliser un système Element backend avec votre installation d'Astra Trident.

Ce dont vous avez besoin

- Système de stockage pris en charge exécutant le logiciel Element.
- Identifiants de locataire ou administrateur de cluster NetApp HCI/SolidFire pouvant gérer les volumes
- Tous vos nœuds workers Kubernetes doivent avoir installé les outils iSCSI appropriés. Voir "[informations de préparation du nœud de travail](#)".

Ce que vous devez savoir

Le `solidfire-san` le pilote de stockage prend en charge les deux modes de volume : fichier et bloc. Pour le `Filesystem` En mode volume, Astra Trident crée un volume et crée un système de fichiers. Le type de système de fichiers est spécifié par la classe de stockage.

Conducteur	Protocole	Mode Volume	Modes d'accès pris en charge	Systèmes de fichiers pris en charge
<code>solidfire-san</code>	ISCSI	Bloc	RWO,ROX,RWX	Aucun système de fichiers. Périphérique de bloc brut.
<code>solidfire-san</code>	ISCSI	Bloc	RWO,ROX,RWX	Aucun système de fichiers. Périphérique de bloc brut.
<code>solidfire-san</code>	ISCSI	Système de fichiers	RWO,ROX	<code>xf</code> s, <code>ext3</code> , <code>ext4</code>
<code>solidfire-san</code>	ISCSI	Système de fichiers	RWO,ROX	<code>xf</code> s, <code>ext3</code> , <code>ext4</code>



Astra Trident utilise le protocole CHAP lorsqu'il fonctionne comme un mécanisme de provisionnement CSI amélioré. Si vous utilisez CHAP (qui est la valeur par défaut pour CSI), aucune autre préparation n'est requise. Il est recommandé de définir explicitement le `UseCHAP` Possibilité d'utiliser CHAP avec Trident non CSI. Sinon, voir "[ici](#)".



Les groupes d'accès aux volumes sont uniquement pris en charge par le framework classique non CSI pour Astra Trident. Lorsqu'il est configuré pour fonctionner en mode CSI, Astra Trident utilise le protocole CHAP.

Si aucun de ces deux cas `AccessGroups` ou `UseCHAP` sont définies, l'une des règles suivantes s'applique :

- Si la valeur par défaut `trident` groupe d'accès détecté, groupes d'accès utilisés.
- Si aucun groupe d'accès n'est détecté et que la version de Kubernetes est 1.7 ou ultérieure, CHAP est utilisé.

Options de configuration du back-end

Voir le tableau suivant pour les options de configuration du back-end :

Paramètre	Description	Valeur par défaut
version		Toujours 1
storageDriverName	Nom du pilote de stockage	Toujours « solidfire-san ».
backendName	Nom personnalisé ou système back-end de stockage	“SolidFire_” + adresse IP de stockage (iSCSI)
Endpoint	MVIP pour le cluster SolidFire avec les identifiants de locataire	
SVIP	Port et adresse IP de stockage (iSCSI)	
labels	Ensemble d'étiquettes arbitraires au format JSON à appliquer aux volumes.	« »
TenantName	Nom du locataire à utiliser (créé si introuvable)	
InitiatorIFace	Limitez le trafic iSCSI à une interface hôte spécifique	« par défaut »
UseCHAP	Utilisez CHAP pour authentifier iSCSI	vrai
AccessGroups	Liste des ID de groupes d'accès à utiliser	Recherche l'ID d'un groupe d'accès nommé « trident »
Types	Spécifications de QoS	
limitVolumeSize	Echec du provisionnement si la taille du volume demandé est supérieure à cette valeur	« » (non appliqué par défaut)
debugTraceFlags	Indicateurs de débogage à utiliser lors du dépannage. Exemple, {“api”:false, “méthode”:true}	nul



Ne pas utiliser `debugTraceFlags` à moins que vous ne soyez en mesure de dépanner et que vous ayez besoin d'un vidage détaillé des journaux.



Pour tous les volumes créés, Astra Trident copie tous les libellés présents dans un pool de stockage vers le LUN de stockage back-end au moment du provisionnement. Les administrateurs de stockage peuvent définir des étiquettes par pool de stockage et regrouper tous les volumes créés dans un pool de stockage. Cela permet de différencier facilement les volumes en fonction d'un ensemble d'étiquettes personnalisables fournies dans la configuration back-end.

Exemple 1 : configuration back-end pour `solidfire-san` avec trois types de volume

Cet exemple montre un fichier back-end utilisant l'authentification CHAP et la modélisation de trois types de

volumes avec des garanties de QoS spécifiques. Il est fort probable que vous définiriez ensuite des classes de stockage pour consommer chacune de ces catégories à l'aide de l' `IOPS` paramètre de classe de stockage.

```
{
  "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 back-end et de la classe de stockage pour `solidfire-san` pilote avec pools de stockage virtuel

Cet exemple représente le fichier de définition back-end configuré avec des pools de stockage virtuel et des classes de stockage qui les renvoient.

Dans l'exemple de fichier de définition de back-end illustré ci-dessous, des valeurs par défaut spécifiques sont définies pour tous les pools de stockage, qui définissent le `type` Du niveau Silver. Les pools de stockage virtuels sont définis dans le `storage` section. 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 de classe de stockage suivantes font référence aux pools de stockage virtuels ci-dessus. À l'aide du `parameters.selector` Chaque classe de stockage indique quel(s) pool(s) virtuel(s) peut(s) être utilisé(s) pour héberger un volume. Les aspects définis dans le pool virtuel sélectionné seront définis pour le volume.

La première classe de stockage (`solidfire-gold-four`) sera mappé sur le premier pool de stockage virtuel. Il s'agit du seul pool offrant des performances Gold avec un `Volume Type QoS` De l'or. La dernière classe de stockage (`solidfire-silver`) appelle n'importe quel pool de stockage qui offre une performance silver. Astra Trident va décider du pool de stockage virtuel sélectionné et s'assurer que les besoins en stockage sont satisfaits.


```
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"
```

Trouvez plus d'informations

- ["Groupes d'accès de volume"](#)

Configurer un système back-end avec les pilotes SAN ONTAP

Découvrez comment configurer un back-end ONTAP avec des pilotes SAN ONTAP ou Cloud Volumes ONTAP.

- ["Préparation"](#)
- ["Configuration et exemples"](#)

Autorisations utilisateur

Astra Trident devrait être exécuté en tant qu'administrateur de ONTAP ou du SVM, généralement à l'aide du `admin` utilisateur du cluster ou un `vsadmin` Utilisateur d'un SVM ou un utilisateur avec un autre nom qui a le même rôle. Pour les déploiements Amazon FSX pour NetApp ONTAP, Astra Trident devrait être exécuté en tant qu'administrateur ONTAP ou SVM, à l'aide du cluster `fsxadmin` utilisateur ou un `vsadmin` Utilisateur d'un SVM ou un utilisateur avec un autre nom qui a le même rôle. Le `fsxadmin` l'utilisateur remplace limitée l'utilisateur administrateur du cluster.



Si vous utilisez le `limitAggregateUsage` paramètre, des autorisations d'administration du cluster sont requises. Avec Amazon FSX pour NetApp ONTAP avec Astra Trident, le `limitAggregateUsage` le paramètre ne fonctionne pas avec le `vsadmin` et `fsxadmin` comptes d'utilisateur. L'opération de configuration échoue si vous spécifiez ce paramètre.

Préparation

Découvrez comment vous préparer à configurer un système ONTAP backend avec les pilotes SAN ONTAP. Pour tous les systèmes back-end ONTAP, Astra Trident requiert au moins un agrégat affecté à la SVM.

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 un `san-dev` classe qui utilise le `ontap-san` conducteur et a `san-default` classe qui utilise le `ontap-san-economy` une seule.

Tous vos nœuds workers Kubernetes doivent avoir installé les outils iSCSI appropriés. Voir ["ici"](#) pour en savoir plus.

Authentification

Astra Trident propose deux modes d'authentification d'un système back-end ONTAP.

- Basé sur les informations d'identification : nom d'utilisateur et mot de passe pour un utilisateur ONTAP disposant des autorisations requises. Il est recommandé d'utiliser un rôle de connexion de sécurité prédéfini, par exemple `admin` ou `vsadmin` Pour garantir une compatibilité maximale avec les versions ONTAP.
- Basé sur des certificats : Astra Trident peut également communiquer avec un cluster ONTAP à l'aide d'un certificat installé sur le système back-end. Dans ce cas, la définition backend doit contenir des valeurs encodées Base64 du certificat client, de la clé et du certificat d'autorité de certification de confiance, le cas échéant (recommandé).

Les utilisateurs ont également la possibilité de mettre à jour les systèmes back-end existants, en choisissant de passer d'un système basé sur les identifiants à un système basé sur les certificats, et inversement. Si **les deux identifiants et certificats sont fournis**, Astra Trident utilisera par défaut les certificats tout en émettant

un avertissement pour supprimer les identifiants de la définition du back-end.

Activer l'authentification basée sur les informations d'identification

Astra Trident nécessite les identifiants d'un administrateur SVM-scoped/cluster-scoped pour communiquer avec le ONTAP backend. Il est recommandé d'utiliser des rôles standard prédéfinis tels que `admin` ou `vsadmin`. Il est ainsi possible d'assurer une compatibilité avec les futures versions d'ONTAP et d'exposer les API de fonctionnalités à utiliser avec les futures versions d'Astra Trident. Un rôle de connexion de sécurité personnalisé peut être créé et utilisé avec Astra Trident, mais il n'est pas recommandé.

Voici un exemple de définition du back-end :

```
{
  "version": 1,
  "backendName": "ExampleBackend",
  "storageDriverName": "ontap-san",
  "managementLIF": "10.0.0.1",
  "dataLIF": "10.0.0.2",
  "svm": "svm_nfs",
  "username": "vsadmin",
  "password": "secret",
}
```

Gardez à l'esprit que la définition du back-end est le seul endroit où les informations d'identification sont stockées en texte brut. Une fois le système backend créé, les noms d'utilisateur/mots de passe sont codés avec Base64 et stockés sous forme de secrets Kubernetes. La création/la conversion d'un back-end est la seule étape qui nécessite la connaissance des informations d'identification. Il s'agit donc d'une opération uniquement administrative, qui doit être effectuée par l'administrateur Kubernetes/du stockage.

Activez l'authentification basée sur les certificats

Les systèmes back-end, nouveaux et existants, peuvent utiliser un certificat et communiquer avec le système back-end ONTAP. Trois paramètres sont requis dans la définition du back-end.

- `ClientCertificate` : valeur encodée en Base64 du certificat client.
- `ClientPrivateKey` : valeur encodée en Base64 de la clé privée associée.
- `TrustedCACertificate` : valeur encodée Base64 du certificat CA de confiance. Si vous utilisez une autorité de certification approuvée, ce paramètre doit être fourni. Ceci peut être ignoré si aucune autorité de certification approuvée n'est utilisée.

Un flux de travail type comprend 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 pour qu'il s'authentifie.

```
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. Il se peut déjà que l'administrateur de stockage gère cet espace. Ignorer si aucune autorité de certification approuvée 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 (à partir 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. Vérifiez que le rôle de connexion de sécurité ONTAP est pris en charge cert 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 d'un certificat généré. Remplacer <ONTAP Management LIF> et <vserver name> par Management LIF IP et SVM name.

```
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 CA 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 back-end à l'aide des valeurs obtenues à partir de l'étape précédente.

```

$ cat cert-backend.json
{
  "version": 1,
  "storageDriverName": "ontap-san",
  "backendName": "SanBackend",
  "managementLIF": "1.2.3.4",
  "dataLIF": "1.2.3.8",
  "svm": "vserver_test",
  "clientCertificate": "Faaaakkkkeeee...Vaaalllluuueeee",
  "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 |
+-----+-----+-----+
+-----+-----+

```

Mettre à jour les méthodes d'authentification ou faire pivoter les informations d'identification

Vous pouvez mettre à jour un back-end existant pour utiliser une méthode d'authentification différente ou pour faire pivoter leurs informations d'identification. Cela fonctionne de deux manières : les systèmes back-end qui utilisent le nom d'utilisateur/mot de passe peuvent être mis à jour pour utiliser des certificats ; les systèmes back-end qui utilisent des certificats peuvent être mis à jour en fonction du nom d'utilisateur/mot de passe. Pour cela, utilisez une mise à jour `backend.json` fichier contenant les paramètres requis à exécuter `tridentctl backend update`.

```

$ cat cert-backend-updated.json
{
  "version": 1,
  "storageDriverName": "ontap-san",
  "backendName": "SanBackend",
  "managementLIF": "1.2.3.4",
  "dataLIF": "1.2.3.8",
  "svm": "vserver_test",
  "username": "vsadmin",
  "password": "secret",
  "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 |
+-----+-----+-----+-----+
+-----+-----+

```



Lors de la rotation 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 back-end. Lors de la rotation de certificats, plusieurs certificats peuvent être ajoutés à l'utilisateur. Le back-end est ensuite mis à jour pour utiliser le nouveau certificat, en suivant lequel l'ancien certificat peut être supprimé du cluster ONTAP.

La mise à jour d'un back-end n'interrompt pas l'accès aux volumes qui ont déjà été créés, et n'a aucun impact sur les connexions de volume effectuées après. Une mise à jour réussie indique qu'Astra Trident peut communiquer avec le système back-end ONTAP et gérer les opérations de volumes à venir.

Spécifiez les igroups

Astra Trident utilise des igroups pour contrôler l'accès aux volumes (LUN) qu'il provisionne. Dans le cas de la spécification des igroups pour un système back-end, les administrateurs ont deux options :

- Astra Trident peut créer et gérer automatiquement un groupe initiateur par système back-end. Si `groupName` n'est pas inclus dans la définition du système back-end, Astra Trident crée un groupe initiateur nommé `trident-<backend-UUID>` Sur le SVM. Cela permet de s'assurer que chaque système back-end dispose d'un groupe initiateur dédié et de gérer l'ajout/la suppression automatiques d'IQN de nœud Kubernetes.

- Alternativement, les groupes pré-cr  s peuvent  tre fournis dans une d finition de back-end. Pour ce faire, utilisez le `igroupName` param tre config. Astra Trident ajoute/supprime des IQN de n ud Kubernetes au groupe initiateur pr existant.

Pour les syst mes back-end dont ils ont besoin `igroupName` d fini, le `igroupName` peut  tre supprim  avec un `tridentctl backend update`. Pour b n ficier des groupes   manipulation automatique avec Astra Trident. L'acc s aux volumes d j  rattach s aux charges de travail ne sera pas perturb . Les futures connexions seront g r es   l'aide du groupe initiateur Astra Trident.



D dier un groupe initiateur   chaque instance unique d'Astra Trident est une bonne pratique b n fique pour l'administrateur Kubernetes et l'administrateur du stockage. CSI Trident automatise l'ajout et la suppression des IQN du n ud du cluster au groupe initiateur, ce qui simplifie consid rablement sa gestion. Lorsque vous utilisez le m me SVM sur tous les environnements Kubernetes (et avec des installations Trident d'Astra), un groupe initiateur d di  permet de s'assurer que les modifications apport es   un cluster Kubernetes n'influencent pas les groupes initiateurs associ s   un autre. En outre, il est important de s'assurer que chaque n ud du cluster Kubernetes dispose d'un IQN unique. Comme mentionn  ci-dessus, Astra Trident s'occupe automatiquement de l'ajout et de la suppression des IQN. La r utilisation d'IQN sur des h tes peut entra ner des sc narios ind sirables o  les h tes se confondent les uns avec les autres et o  l'acc s aux LUN est refus .

Si Astra Trident est configur  pour fonctionner comme un provisionnement CSI, les IQN du n ud Kubernetes sont automatiquement ajout s ou supprim s du groupe initiateur. Lorsque des n uds sont ajout s   un cluster Kubernetes, `trident-csi` DemonSet d ploie un pod (`trident-csi-xxxxx`) sur les n uds r cemment ajout s et enregistre les nouveaux n uds sur lesquels il peut attacher des volumes. Les IQN du n ud sont  galement ajout s au groupe initiateur du back-end. Un ensemble d' tapes similaire g re la suppression des IQN lorsque le(s) n ud(s) est cordeleted, drain  et supprim  de Kubernetes.

Si Astra Trident ne s'ex cute pas comme un provisionnement CSI, le groupe initiateur doit  tre mis   jour manuellement pour contenir les IQN iSCSI de chaque n ud worker du cluster Kubernetes. Les IQN des n uds qui rejoignent le cluster Kubernetes devront  tre ajout s au groupe initiateur. De m me, les IQN des n uds qui sont supprim s du cluster Kubernetes doivent  tre supprim s du groupe initiateur.

Authentifier les connexions avec le protocole CHAP bidirectionnel

Astra Trident peut authentifier les sessions iSCSI avec le protocole CHAP bidirectionnel pour le `ontap-san` et `ontap-san-economy` pilotes. Pour cela, il faut activer `useCHAP` dans votre d finition backend. Lorsqu'il est r gl  sur `true`, Astra Trident configure la s curit  de l'initiateur par d faut du SVM en CHAP bidirectionnel et d finit le nom d'utilisateur et les secrets du fichier backend. NetApp recommande d'utiliser le protocole CHAP bidirectionnel pour l'authentification des 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": "FaKePaSsWoRd",
  "igroupName": "trident",
  "chapInitiatorSecret": "cl9qxIm36DKyawxy",
  "chapTargetInitiatorSecret": "rqxigXgkesIpwxyz",
  "chapTargetUsername": "iJF4heBRT0TCwxyz",
  "chapUsername": "uh2aNCLSD6cNwxyz",
}

```



Le `useCHAP` Paramètre est une option booléenne qui ne peut être configurée qu'une seule fois. Elle est définie sur `FALSE` par défaut. Une fois la valeur `true` définie, vous ne pouvez pas la définir sur `false`.

En plus de `useCHAP=true`, le `chapInitiatorSecret`, `chapTargetInitiatorSecret`, `chapTargetUsername`, et `chapUsername` les champs doivent être inclus dans la définition back-end. Les secrets peuvent être modifiés après la création d'un back-end en cours d'exécution `tridentctl update`.

Comment cela fonctionne

Par réglage `useCHAP` À vrai dire, l'administrateur du stockage demande à Astra Trident de configurer le protocole CHAP sur le système back-end. Ceci inclut les éléments suivants :

- Configuration du protocole CHAP sur le SVM :
 - Si le type de sécurité de l'initiateur par défaut du SVM n'est pas défini (défini par défaut) **et** il n'y a pas de LUN préexistantes dans le volume, Astra Trident définit le type de sécurité par défaut sur `CHAP` Et procédez à la configuration de l'initiateur CHAP et du nom d'utilisateur cible et des secrets.
 - Si le SVM contient des LUN, Astra Trident n'active pas le protocole CHAP sur le SVM. Cela permet de garantir que l'accès aux LUN déjà présentes sur le SVM n'est pas restreint.
- Configuration de l'initiateur CHAP et du nom d'utilisateur cible et des secrets ; ces options doivent être spécifiées dans la configuration backend (comme indiqué ci-dessus).
- Gestion de l'ajout d'initiateurs au `igroupName` donné en arrière-plan. Si ce n'est pas spécifié, la valeur par défaut est `trident`.

Une fois le système back-end créé, Astra Trident crée un correspondant `tridentbackend` CRD et stocke les secrets et noms d'utilisateur CHAP sous forme de secrets Kubernetes. Tous les volumes persistants créés par Astra Trident sur ce back-end seront montés et rattachés au protocole CHAP.

Rotation des identifiants et mise à jour des systèmes back-end

Vous pouvez mettre à jour les informations d'identification CHAP en mettant à jour les paramètres CHAP dans

le `backend.json` fichier. Cela nécessitera la mise à jour des secrets CHAP et l'utilisation de `tridentctl update` pour refléter ces modifications.



Lors de la mise à jour des secrets CHAP pour un back-end, vous devez utiliser `tridentctl` pour mettre à jour le backend. Ne mettez pas à jour les identifiants du cluster de stockage via l'interface de ligne de commande/ONTAP car Astra Trident ne pourra pas détecter 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": "FaKePaSsWoRd",
  "igroupName": "trident",
  "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 ne seront pas affectées. Elles restent actives si les identifiants sont mis à jour par Astra Trident sur le SVM. Les nouvelles connexions utiliseront les informations d'identification mises à jour et les connexions existantes continuent de rester actives. La déconnexion et la reconnexion des anciens volumes persistants se traduiront par l'utilisation des identifiants mis à jour.

Exemples et options de configuration

Découvrez comment créer et utiliser des pilotes SAN ONTAP avec votre installation d'Astra Trident. Cette section présente des exemples de configuration du back-end et des détails sur le mappage des systèmes

back-end aux classes de stockage.

Options de configuration du back-end

Voir le tableau suivant pour les options de configuration du back-end :

Paramètre	Description	Valeur par défaut
version		Toujours 1
storageDriverName	Nom du pilote de stockage	ontap-nas, ontap-nas-économie, ontap-nas-flexgroup, ontap-san », « ontap-san », « ontap-économie san »
backendName	Nom personnalisé ou système back-end de stockage	Nom du pilote + "_" + dataLIF
managementLIF	Adresse IP d'un cluster ou d'une LIF de gestion SVM	« 10.0.0.1 », « [2001:1234:abcd::fefe] »
dataLIF	Adresse IP de la LIF de protocole. Utilisez des crochets pour IPv6. Ne peut pas être mis à jour une fois que vous l'avez défini	Dérivé par la SVM sauf spécification
useCHAP	Utiliser CHAP pour authentifier iSCSI pour les pilotes SAN ONTAP [Boolean]	faux
chapInitiatorSecret	Secret de l'initiateur CHAP. Requis si useCHAP=true	« »
labels	Ensemble d'étiquettes arbitraires au format JSON à appliquer aux volumes	« »
chapTargetInitiatorSecret	Secret de l'initiateur cible CHAP. Requis si useCHAP=true	« »
chapUsername	Nom d'utilisateur entrant. Requis si useCHAP=true	« »
chapTargetUsername	Nom d'utilisateur cible. Requis 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 CA de confiance. Facultatif. Utilisé pour l'authentification par certificat	« »

Paramètre	Description	Valeur par défaut
username	Nom d'utilisateur pour la connexion au cluster/SVM. Utilisé pour l'authentification basée sur les identifiants	« »
password	Mot de passe pour la connexion au cluster/SVM. Utilisé pour l'authentification basée sur les identifiants	« »
svm	Serveur virtuel de stockage à utiliser	Dérivé d'un SVM managementLIF est spécifié
igroupName	Nom du groupe initiateur à utiliser pour les volumes SAN	Trident-<backend-UUID>
storagePrefix	Préfixe utilisé pour le provisionnement des nouveaux volumes dans la SVM. Ne peut pas être mis à jour une fois que vous l'avez défini	trident
limitAggregateUsage	Echec du provisionnement si l'utilisation est supérieure à ce pourcentage. Ne s'applique pas à Amazon FSX pour ONTAP	« » (non appliqué par défaut)
limitVolumeSize	Echec du provisionnement si la taille du volume demandé est supérieure à cette valeur.	« » (non appliqué par défaut)
lunsPerFlexvol	Nombre maximal de LUN par FlexVol, doit être compris dans la plage [50, 200]	"100"
debugTraceFlags	Indicateurs de débogage à utiliser lors du dépannage. Exemple, {"api":false, "méthode":true}	nul
useREST	Paramètre booléen pour utiliser les API REST de ONTAP. Aperçu technique	faux



useREST est fourni sous forme d'aperçu technique ** qui est recommandé pour les environnements de test et non pour les charges de travail de production. Lorsqu'il est réglé sur true, Astra Trident va utiliser les API REST de ONTAP pour communiquer avec le système back-end. Cette fonctionnalité requiert ONTAP 9.9 et versions ultérieures. En outre, le rôle de connexion ONTAP utilisé doit avoir accès au `ontap` client supplémentaire. Ceci est satisfait par le pré-défini `vsadmin` et `cluster-admin` rôles.

Pour communiquer avec le cluster ONTAP, vous devez fournir les paramètres d'authentification. Il peut s'agir du nom d'utilisateur/mot de passe d'une connexion de sécurité ou d'un certificat installé.



Si vous utilisez un système Amazon FSX pour le système back-end NetApp ONTAP, ne spécifiez pas le système `limitAggregateUsage` paramètre. Le `fsxadmin` et `vsadmin` Les rôles fournis par Amazon FSX pour NetApp ONTAP ne contiennent pas les autorisations d'accès requises pour récupérer l'utilisation des agrégats et le limiter via Astra Trident.



Ne pas utiliser `debugTraceFlags` à moins que vous ne soyez en mesure de déboguer et que vous ayez besoin d'un vidage détaillé des journaux.

Pour le `ontap-san` Pilotes, par défaut est d'utiliser toutes les adresses IP des LIF de données du SVM et d'utiliser le chemin d'accès multivoie iSCSI. Spécification d'une adresse IP pour la LIF de données pour le `ontap-san` les pilotes les obligent à désactiver le multichemin et à utiliser uniquement l'adresse spécifiée.



Lors de la création d'un back-end, n'oubliez pas que `dataLIF` et `storagePrefix` ne peut pas être modifié après sa création. Pour mettre à jour ces paramètres, vous devez créer un nouveau back-end.

`igroupName` Peut être défini sur un groupe initiateur déjà créé sur le cluster ONTAP. Si non spécifié, Astra Trident crée automatiquement un groupe initiateur nommé `trident-<back-end-UUID>`. Si l'on fournit un nom de partenaire prédéfini, NetApp recommande d'utiliser un groupe initiateur par cluster Kubernetes si le SVM doit être partagé entre les environnements. Cela est nécessaire pour qu'Astra Trident conserve automatiquement les ajouts/suppressions d'IQN.

Les systèmes back-end peuvent également avoir mis à jour les groupes initiateurs après leur création :

- Vous pouvez mettre à jour le nom de l'outil afin de désigner un nouveau groupe initiateur créé et géré sur la SVM en dehors d'Astra Trident.
- Le nom de l'utilisateur peut être omis. Dans ce cas, Astra Trident crée et gère automatiquement un groupe initiateur `trident-<back-end-UUID>`.

Dans les deux cas, les pièces jointes de volume continueront d'être accessibles. Les pièces jointes futures utilisent le groupe initiateur mis à jour. Cette mise à jour n'interrompt pas l'accès aux volumes présents sur le back-end.

Un nom de domaine complet (FQDN) peut être spécifié pour le `managementLIF` option.

```
`managementLIF` Pour tous les pilotes ONTAP peuvent également être définis sur des adresses IPv6. Veillez à installer Trident avec le `--use-ipv6` drapeau. Il faut veiller à définir `managementLIF` Adresse IPv6 entre crochets.
```



Lorsque vous utilisez des adresses IPv6, assurez-vous de `managementLIF` et `dataLIF` (si inclus dans votre définition de back-end) sont définis entre crochets, tels que `[28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]`. Si `dataLIF` N'est pas fourni, Astra Trident va récupérer les LIF de données IPv6 à partir du SVM.

Pour activer les pilotes `ontap-san` à l'aide du protocole CHAP, définissez la `useCHAP` paramètre à `true` dans votre définition de back-end. Astra Trident configure ensuite et utilise le protocole CHAP bidirectionnel comme authentification par défaut pour la SVM donnée en back-end. Voir "[ici](#)" pour en savoir plus sur son fonctionnement.

Pour le `ontap-san-economy` conducteur, le `limitVolumeSize` Elle limite également la taille maximale des volumes qu'elle gère pour les qtrees et les LUN.



Astra Trident définit les libellés de provisionnement dans le champ « Commentaires » de tous les volumes créés à l'aide de l' `ontap-san` conducteur. Pour chaque volume créé, le champ « Commentaires » de la FlexVol est rempli avec toutes les étiquettes présentes sur le pool de stockage dans lequel elle est placée. Les administrateurs de stockage peuvent définir des étiquettes par pool de stockage et regrouper tous les volumes créés dans un pool de stockage. Cela permet de différencier facilement les volumes en fonction d'un ensemble d'étiquettes personnalisables fournies dans la configuration back-end.

Options de configuration back-end pour les volumes de provisionnement

Vous pouvez contrôler la façon dont chaque volume est provisionné par défaut à l'aide de ces options dans une section spéciale de la configuration. Pour un exemple, voir les exemples de configuration ci-dessous.

Paramètre	Description	Valeur par défaut
<code>spaceAllocation</code>	Allocation d'espace pour les LUN	« vrai »
<code>spaceReserve</code>	Mode de réservation d'espace ; "none" (fin) ou "volume" (épais)	« aucun »
<code>snapshotPolicy</code>	Règle Snapshot à utiliser	« aucun »
<code>qosPolicy</code>	QoS policy group à affecter pour les volumes créés. Choisissez une de <code>qosPolicy</code> ou <code>adaptiveQosPolicy</code> par pool de stockage/back-end	« »
<code>adaptiveQosPolicy</code>	Groupe de règles de QoS adaptative à attribuer aux volumes créés. Choisissez une de <code>qosPolicy</code> ou <code>adaptiveQosPolicy</code> par pool de stockage/back-end	« »
<code>snapshotReserve</code>	Pourcentage du volume réservé pour les instantanés "0"	Si <code>snapshotPolicy</code> est « aucun », sinon « »
<code>splitOnClone</code>	Séparer un clone de son parent lors de sa création	« faux »
<code>splitOnClone</code>	Séparer un clone de son parent lors de sa création	« faux »
<code>encryption</code>	Activer le chiffrement de volume NetApp	« faux »
<code>securityStyle</code>	Style de sécurité pour les nouveaux volumes	"unix"
<code>tieringPolicy</code>	La stratégie de hiérarchisation à utiliser « none »	Snapshot uniquement pour une configuration SVM-DR pré-ONTAP 9.5



Avec Astra Trident, les groupes de règles de QoS doivent être utilisés avec ONTAP 9.8 ou version ultérieure. Il est recommandé d'utiliser un groupe de règles de qualité de service non partagé et de s'assurer que le groupe de règles est appliqué à chaque composant individuellement. Un groupe de règles de QoS partagé appliquera le plafond du débit total de toutes les charges de travail.

Voici un exemple avec des valeurs par défaut définies :

```
{
  "version": 1,
  "storageDriverName": "ontap-san",
  "managementLIF": "10.0.0.1",
  "dataLIF": "10.0.0.2",
  "svm": "trident_svm",
  "username": "admin",
  "password": "password",
  "labels": {"k8scluster": "dev2", "backend": "dev2-sanbackend"},
  "storagePrefix": "alternate-trident",
  "igroupName": "custom",
  "debugTraceFlags": {"api":false, "method":true},
  "defaults": {
    "spaceReserve": "volume",
    "qosPolicy": "standard",
    "spaceAllocation": "false",
    "snapshotPolicy": "default",
    "snapshotReserve": "10"
  }
}
```



Pour tous les volumes créés à l'aide de `ontap-san` Avec d'autres pilotes, Astra Trident ajoute une capacité supplémentaire de 10 % au système FlexVol pour prendre en charge les métadonnées de LUN. La LUN sera provisionnée avec la taille exacte que l'utilisateur demande dans la demande de volume persistant. Astra Trident ajoute 10 % au système FlexVol (dont la taille disponible dans ONTAP). Les utilisateurs obtiennent à présent la capacité utilisable requise. Cette modification empêche également que les LUN ne soient en lecture seule, à moins que l'espace disponible soit pleinement utilisé. Cela ne s'applique pas à l'économie d'`ontap-san`.

Pour les systèmes back-end définis `snapshotReserve`, Astra Trident calcule la taille des volumes comme suit :

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

Le modèle 1.1 est le modèle 10 % d'Astra Trident supplémentaire qui s'ajoute à la baie FlexVol pour prendre en charge les métadonnées de la LUN. Pour `snapshotReserve` = 5 % et demande de volume persistant = 5

Gio, la taille totale du volume est de 5,7 Gio et la taille disponible est de 5,5 Gio. Le volume show la commande doit afficher des résultats similaires à cet exemple :

Vserver	Volume	Aggregate	State	Type	Size	Available	Used%
		_pvc_89f1c156_3801_4de4_9f9d_034d54c395f4	online	RW	10GB	5.00GB	0%
		_pvc_e42ec6fe_3baa_4af6_996d_134adbbb8e6d	online	RW	5.79GB	5.50GB	0%
		_pvc_e8372153_9ad9_474a_951a_08ae15e1c0ba	online	RW	1GB	511.8MB	0%

3 entries were displayed.

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

Exemples de configuration minimaux

Les exemples suivants montrent des configurations de base qui laissent la plupart des paramètres par défaut. C'est la façon la plus simple de définir un back-end.



Si vous utilisez Amazon FSX sur NetApp ONTAP avec Astra Trident, il est recommandé de spécifier des noms DNS pour les LIF au lieu d'adresses IP.

ontap-san pilote avec authentification par certificat

Il s'agit d'un exemple de configuration back-end minimal. `clientCertificate`, `clientPrivateKey`, et `trustedCACertificate` (Facultatif, si vous utilisez une autorité de certification approuvée) est renseigné `backend.json` Et prendre les valeurs codées en base64 du certificat client, de la clé privée et du certificat CA de confiance, respectivement.

```
{
  "version": 1,
  "storageDriverName": "ontap-san",
  "backendName": "DefaultSANBackend",
  "managementLIF": "10.0.0.1",
  "dataLIF": "10.0.0.3",
  "svm": "svm_iscsi",
  "useCHAP": true,
  "chapInitiatorSecret": "cl9qxIm36DKyawxy",
  "chapTargetInitiatorSecret": "rqxigXgkesIpwxyz",
  "chapTargetUsername": "iJF4heBRT0TCwxyz",
  "chapUsername": "uh2aNCLSD6cNwxyz",
  "igroupName": "trident",
  "clientCertificate": "ZXR0ZXJwYXB...ICMgJ3BhcGVyc2",
  "clientPrivateKey": "vciwKIyAgZG...0cnksIGRlc2NyaX",
  "trustedCACertificate": "zcyBbaG...b3Igb3duIGNsYXNz"
}
```

ontap-san **Pilote avec CHAP bidirectionnel**

Il s'agit d'un exemple de configuration back-end minimal. Cette configuration de base crée un `ontap-san` backend avec `useCHAP` réglé sur `true`.

```
{
  "version": 1,
  "storageDriverName": "ontap-san",
  "managementLIF": "10.0.0.1",
  "dataLIF": "10.0.0.3",
  "svm": "svm_iscsi",
  "labels": {"k8scluster": "test-cluster-1", "backend": "testcluster1-
sanbackend"},
  "useCHAP": true,
  "chapInitiatorSecret": "cl9qxIm36DKyawxy",
  "chapTargetInitiatorSecret": "rqxigXgkesIpwxyz",
  "chapTargetUsername": "iJF4heBRT0TCwxyz",
  "chapUsername": "uh2aNCLSD6cNwxyz",
  "igroupName": "trident",
  "username": "vsadmin",
  "password": "secret"
}
```

ontap-san-economy **conducteur**

```
{
  "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",
  "igroupName": "trident",
  "username": "vsadmin",
  "password": "secret"
}
```

Exemples de systèmes back-end avec pools de stockage virtuel

Dans l'exemple de fichier de définition backend ci-dessous, des valeurs par défaut spécifiques sont définies pour tous les pools de stockage, par exemple `spaceReserve` aucune, `spaceAllocation` lors de la fausse idée, et `encryption` faux. Les pools de stockage virtuels sont définis dans la section `stockage`.

Dans cet exemple, certains pools de stockage sont propriétaires de leur propre pool `spaceReserve`, `spaceAllocation`, et `encryption` les valeurs et certains pools remplacent les valeurs par défaut définies ci-dessus.

```
{
  "version": 1,
  "storageDriverName": "ontap-san",
  "managementLIF": "10.0.0.1",
  "dataLIF": "10.0.0.3",
  "svm": "svm_iscsi",
  "useCHAP": true,
  "chapInitiatorSecret": "cl9qxIm36DKyawxy",
  "chapTargetInitiatorSecret": "rqxigXgkesIpwxyz",
  "chapTargetUsername": "iJF4heBRT0TCwxyz",
  "chapUsername": "uh2aNCLSD6cNwxyz",
  "igroupName": "trident",
  "username": "vsadmin",
  "password": "secret",

  "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"
      }
    }
  ]
}

```

Voici un exemple iSCSI pour le ontap-san-economy pilote :

```

{
  "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",
  "igroupName": "trident",
  "username": "vsadmin",
  "password": "secret",

  "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"
        }
    }
]
}

```

Mapping des systèmes back-end aux classes de stockage

Les définitions de classe de stockage suivantes font référence aux pools de stockage virtuels ci-dessus. À l'aide du `parameters.selector` Chaque classe de stockage indique quel(s) pool(s) virtuel(s) peut(s) être utilisé(s) pour héberger un volume. Les aspects définis dans le pool virtuel sélectionné seront définis pour le volume.

- La première classe de stockage (`protection-gold`) sera mappé sur le premier, deuxième pool de stockage virtuel dans le `ontap-nas-flexgroup` système back-end et le premier pool de stockage virtuel dans `ontap-san` back-end. Il s'agit du seul pool offrant une protection de niveau Gold.
- La deuxième classe de stockage (`protection-not-gold`) sera mappé sur le troisième, quatrième pool de stockage virtuel dans `ontap-nas-flexgroup` back-end et le deuxième, troisième pool de stockage virtuel dans `ontap-san` back-end. Ce sont les seuls pools offrant un niveau de protection autre que l'or.
- La troisième classe de stockage (`app-mysqldb`) sera mappé sur le quatrième pool de stockage virtuel dans `ontap-nas` back-end et le troisième pool de stockage virtuel dans `ontap-san-economy` back-end. Ce sont les seuls pools offrant une configuration de pool de stockage pour l'application de type `mysqldb`.
- La quatrième classe de stockage (`protection-silver-creditpoints-20k`) sera mappé sur le troisième pool de stockage virtuel dans `ontap-nas-flexgroup` back-end et le second pool de stockage virtuel dans `ontap-san` back-end. Ce sont les seules piscines offrant une protection de niveau or à 20000 points de solvabilité.
- La cinquième classe de stockage (`creditpoints-5k`) sera mappé sur le second pool de stockage virtuel dans `ontap-nas-economy` back-end et le troisième pool de stockage virtuel dans `ontap-san` back-end. Ce sont les seules offres de piscine à 5000 points de solvabilité.

Astra Trident va décider du pool de stockage virtuel sélectionné et s'assurer que les besoins en stockage sont satisfaits.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-gold
provisioner: netapp.io/trident
parameters:
  selector: "protection=gold"
  fsType: "ext4"
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-not-gold
provisioner: netapp.io/trident
parameters:
  selector: "protection!=gold"
  fsType: "ext4"
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: app-mysqldb
provisioner: netapp.io/trident
parameters:
  selector: "app=mysqldb"
  fsType: "ext4"
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-silver-creditpoints-20k
provisioner: netapp.io/trident
parameters:
  selector: "protection=silver; creditpoints=20000"
  fsType: "ext4"
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: creditpoints-5k
provisioner: netapp.io/trident
parameters:
  selector: "creditpoints=5000"
  fsType: "ext4"
```

Configurer un système back-end avec les pilotes NAS ONTAP

Découvrez comment configurer un back-end ONTAP avec des pilotes ONTAP ou NAS Cloud Volumes ONTAP.

- ["Préparation"](#)
- ["Configuration et exemples"](#)

Autorisations utilisateur

Astra Trident devrait être exécuté en tant qu'administrateur de ONTAP ou du SVM, généralement à l'aide du `admin` utilisateur du cluster ou un `vsadmin` Utilisateur d'un SVM ou un utilisateur avec un autre nom qui a le même rôle. Pour les déploiements Amazon FSX pour NetApp ONTAP, Astra Trident devrait être exécuté en tant qu'administrateur ONTAP ou SVM, à l'aide du cluster `fsxadmin` utilisateur ou un `vsadmin` Utilisateur d'un SVM ou un utilisateur avec un autre nom qui a le même rôle. Le `fsxadmin` l'utilisateur remplace limitée l'utilisateur administrateur du cluster.



Si vous utilisez le `limitAggregateUsage` paramètre, des autorisations d'administration du cluster sont requises. Avec Amazon FSX pour NetApp ONTAP avec Astra Trident, le `limitAggregateUsage` le paramètre ne fonctionne pas avec le `vsadmin` et `fsxadmin` comptes d'utilisateur. L'opération de configuration échoue si vous spécifiez ce paramètre.

Préparation

Découvrez comment vous préparer à configurer un back-end ONTAP avec les pilotes NAS ONTAP. Pour tous les systèmes back-end ONTAP, Astra Trident requiert au moins un agrégat affecté à la SVM.

Pour tous les systèmes back-end ONTAP, Astra Trident requiert au moins un agrégat affecté à la SVM.

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 classe Gold qui utilise le `ontap-nas` Pilote et une classe Bronze qui utilise le `ontap-nas-economy` une seule.

Tous vos nœuds workers Kubernetes doivent avoir installé les outils NFS appropriés. Voir ["ici"](#) pour en savoir plus.

Authentification

Astra Trident propose deux modes d'authentification d'un système back-end ONTAP.

- Basé sur les informations d'identification : nom d'utilisateur et mot de passe pour un utilisateur ONTAP disposant des autorisations requises. Il est recommandé d'utiliser un rôle de connexion de sécurité prédéfini, par exemple `admin` ou `vsadmin` Pour garantir une compatibilité maximale avec les versions ONTAP.
- Basé sur des certificats : Astra Trident peut également communiquer avec un cluster ONTAP à l'aide d'un certificat installé sur le système back-end. Dans ce cas, la définition backend doit contenir des valeurs encodées Base64 du certificat client, de la clé et du certificat d'autorité de certification de confiance, le cas échéant (recommandé).

Les utilisateurs ont également la possibilité de mettre à jour les systèmes back-end existants, en choisissant de passer d'un système basé sur les identifiants à un système basé sur les certificats, et inversement. Si **les deux identifiants et certificats sont fournis**, Astra Trident utilisera par défaut les certificats tout en émettant un avertissement pour supprimer les identifiants de la définition du back-end.

Activer l'authentification basée sur les informations d'identification

Astra Trident nécessite les identifiants d'un administrateur SVM-scoped/cluster-scoped pour communiquer avec le ONTAP backend. Il est recommandé d'utiliser des rôles standard prédéfinis tels que `admin` ou `vsadmin`. Il est ainsi possible d'assurer une compatibilité avec les futures versions d'ONTAP et d'exposer les API de fonctionnalités à utiliser avec les futures versions d'Astra Trident. Un rôle de connexion de sécurité personnalisé peut être créé et utilisé avec Astra Trident, mais il n'est pas recommandé.

Voici un exemple de définition du back-end :

```
{
  "version": 1,
  "backendName": "ExampleBackend",
  "storageDriverName": "ontap-nas",
  "managementLIF": "10.0.0.1",
  "dataLIF": "10.0.0.2",
  "svm": "svm_nfs",
  "username": "vsadmin",
  "password": "secret"
}
```

Gardez à l'esprit que la définition du back-end est le seul endroit où les informations d'identification sont stockées en texte brut. Une fois le système backend créé, les noms d'utilisateur/mots de passe sont codés avec Base64 et stockés sous forme de secrets Kubernetes. La création/la conversion d'un back-end est la seule étape qui nécessite la connaissance des informations d'identification. Il s'agit donc d'une opération uniquement administrative, qui doit être effectuée par l'administrateur Kubernetes/du stockage.

Activez l'authentification basée sur les certificats

Les systèmes back-end, nouveaux et existants, peuvent utiliser un certificat et communiquer avec le système back-end ONTAP. Trois paramètres sont requis dans la définition du back-end.

- `ClientCertificate` : valeur encodée en Base64 du certificat client.
- `ClientPrivateKey` : valeur encodée en Base64 de la clé privée associée.
- `TrustedCACertificate` : valeur encodée Base64 du certificat CA de confiance. Si vous utilisez une autorité de certification approuvée, ce paramètre doit être fourni. Ceci peut être ignoré si aucune autorité de certification approuvée n'est utilisée.

Un flux de travail type comprend 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 pour qu'il s'authentifie.

```
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. Il se peut déjà que

l'administrateur de stockage gère cet espace. Ignorer si aucune autorité de certification approuvée 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 (à partir 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. Vérifiez que le rôle de connexion de sécurité ONTAP est pris en charge cert methode 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 d'un certificat généré. Remplacer <ONTAP Management LIF> et <vserver name> par Management LIF IP et SVM name. Vous devez vous assurer que le LIF a sa politique 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 CA 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 back-end à l'aide des valeurs obtenues à partir de 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...Vaaalllluuueeee",
  "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 |
+-----+-----+-----+-----+
+-----+-----+

```

Mettre à jour les méthodes d'authentification ou faire pivoter les informations d'identification

Vous pouvez mettre à jour un back-end existant pour utiliser une méthode d'authentification différente ou pour faire pivoter leurs informations d'identification. Cela fonctionne de deux manières : les systèmes back-end qui utilisent le nom d'utilisateur/mot de passe peuvent être mis à jour pour utiliser des certificats ; les systèmes back-end qui utilisent des certificats peuvent être mis à jour en fonction du nom d'utilisateur/mot de passe. Pour cela, utilisez une mise à jour `backend.json` fichier contenant les paramètres requis à exécuter `tridentctl backend update`.


```

$ 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": "secret",
"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 |
+-----+-----+-----+-----+
+-----+-----+

```



Lors de la rotation 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 back-end. Lors de la rotation de certificats, plusieurs certificats peuvent être ajoutés à l'utilisateur. Le back-end est ensuite mis à jour pour utiliser le nouveau certificat, en suivant lequel l'ancien certificat peut être supprimé du cluster ONTAP.

La mise à jour d'un back-end n'interrompt pas l'accès aux volumes qui ont déjà été créés, et n'a aucun impact sur les connexions de volume effectuées après. Une mise à jour réussie indique qu'Astra Trident peut communiquer avec le système back-end ONTAP et gérer les opérations de volumes à venir.

Gestion des règles d'exportation NFS

Astra Trident utilise les règles d'exportation NFS pour contrôler l'accès aux volumes qu'il provisionne.

Astra Trident propose deux options pour l'utilisation des règles d'exportation :

- Astra Trident peut gérer la règle d'exportation de manière dynamique. Dans ce mode de fonctionnement, l'administrateur du stockage spécifie une liste de blocs CIDR qui représentent les adresses IP admissibles. Astra Trident ajoute automatiquement des adresses IP de nœud qui font partie de ces plages à la règle d'exportation. En outre, lorsqu'aucun CIDRS n'est spécifié, toute adresse IP unicast globale trouvée sur les nœuds est ajoutée à la règle d'exportation.

- Les administrateurs du stockage peuvent créer une export-policy et ajouter des règles manuellement. Astra Trident utilise la export policy par défaut, sauf si un nom différent de export policy est spécifié dans la configuration.

Gérez les règles d'exportation de manière dynamique

La version 20.04 de CSI Trident permet de gérer de manière dynamique les règles d'exportation pour les systèmes back-end ONTAP. Cela permet à l'administrateur du stockage de spécifier un espace d'adresse autorisé pour les adresses IP du nœud de travail, au lieu de définir manuellement des règles explicites. Il simplifie considérablement la gestion des export policy ; les modifications apportées à l'export policy ne nécessitent plus d'intervention manuelle sur le cluster de stockage. De plus, cela limite l'accès au cluster de stockage uniquement aux nœuds workers dont les adresses IP sont comprises dans la plage spécifiée, ce qui permet une gestion automatisée et précise.



La gestion dynamique des règles d'exportation n'est disponible que pour CSI Trident. Il est important de s'assurer que les nœuds de travail ne sont pas NATed.

Exemple

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

```
{
  "version": 1,
  "storageDriverName": "ontap-nas",
  "backendName": "ontap_nas_auto_export",
  "managementLIF": "192.168.0.135",
  "svm": "svm1",
  "username": "vsadmin",
  "password": "FaKePaSsWoRd",
  "autoExportCIDRs": ["192.168.0.0/24"],
  "autoExportPolicy": true
}
```



Lorsque vous utilisez cette fonctionnalité, vous devez vous assurer que la jonction root de votre SVM possède une export policy préétablie avec une règle d'exportation qui permet le bloc CIDR du nœud (comme la export policy par défaut). Suivez toujours la meilleure pratique recommandée par NetApp pour dédier un SVM à Astra Trident.

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

- `autoExportPolicy` est défini sur `true`. Cela signifie qu'Astra Trident va créer une export policy pour le `svm1` SVM et gère l'ajout et la suppression de règles à l'aide de `autoExportCIDRs` blocs d'adresse. Par exemple, un backend avec UUID `403b5326-8482-40db-96d0-d83fb3f4daec` et `autoExportPolicy` réglé sur `true` crée une export-policy nommée `trident-403b5326-8482-40db-96d0-d83fb3f4daec` Sur le SVM.
- `autoExportCIDRs` contient une liste de blocs d'adresses. Ce champ est facultatif et il prend par défaut la valeur `["0.0.0.0/0", "*/0"]`. S'il n'est pas défini, Astra Trident ajoute toutes les adresses de diffusion individuelle à périmètre global présentes sur les nœuds du worker.

Dans cet exemple, le 192.168.0.0/24 l'espace d'adressage est fourni. Cela indique que les adresses IP des nœuds Kubernetes qui appartiennent à cette plage d'adresse seront ajoutées à la règle d'exportation créée par Astra Trident. Lorsque Astra Trident enregistre un nœud sur lequel il s'exécute, il récupère les adresses IP du nœud et les vérifie par rapport aux blocs d'adresse fournis dans `autoExportCIDRs`. Après avoir filtré les adresses IP, Astra Trident crée des règles de politique d'exportation pour les adresses IP clientes qu'il détecte, avec une règle pour chaque nœud qu'il identifie.

Vous pouvez mettre à jour `autoExportPolicy` et `autoExportCIDRs` pour les systèmes back-end après leur création. Vous pouvez ajouter de nouveaux rapports CIDR pour un back-end qui est géré automatiquement ou supprimé des rapports CIDR existants. Faites preuve de prudence lors de la suppression des CIDR pour vous assurer que les connexions existantes ne sont pas tombées. Vous pouvez également choisir de désactiver `autoExportPolicy` pour un back-end et revenir à une export policy créée manuellement. Pour ce faire, vous devrez définir le `exportPolicy` dans votre configuration backend.

Après la création ou la mise à jour d'Astra Trident, vous pouvez vérifier le système back-end à l'aide de `tridentctl` ou le correspondant `tridentbackend` CRD :

```
$ ./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>
      fileType: ext4
```

Lorsque des nœuds sont ajoutés à un cluster Kubernetes et enregistrés avec le contrôleur Trident Astra, les règles d'exportation des systèmes back-end existants sont mises à jour (à condition qu'elles tombent dans la plage d'adresse spécifiée dans la `autoExportCIDRs` pour le back-end).

Lorsqu'un nœud est retiré, Astra Trident vérifie tous les systèmes back-end en ligne afin de supprimer la règle d'accès du nœud. En supprimant cette IP de nœud des règles d'exportation des systèmes back-end gérés, Astra Trident empêche les montages erratiques, à moins que cette adresse IP soit réutilisée par un nouveau nœud du cluster.

Pour les systèmes back-end existants, mise à jour du système back-end avec `tridentctl update backend` S'assure qu'Astra Trident gère automatiquement les règles d'exportation. Cela créera une nouvelle

export policy nommée après l'UUID et les volumes du backend qui sont présents sur le back-end, utilisera la export policy nouvellement créée lorsqu'ils sont de nouveau montés.



La suppression d'un back-end avec des règles d'exportation gérées automatiquement supprimera l'export policy créée de manière dynamique. Si le back-end est recréés, il est traité comme un nouveau backend et entraîne la création d'une nouvelle export policy.

Si l'adresse IP d'un nœud actif est mise à jour, vous devez redémarrer le pod Astra Trident sur le nœud. Astra Trident va ensuite mettre à jour la règle d'exportation pour les systèmes back-end qu'il gère pour tenir compte de ce changement d'IP.

Exemples et options de configuration

Découvrez comment créer et utiliser des pilotes NAS ONTAP avec votre installation d'Astra Trident. Cette section présente des exemples de configuration du back-end et des détails sur le mappage des systèmes back-end aux classes de stockage.

Options de configuration du back-end

Voir le tableau suivant pour les options de configuration du back-end :

Paramètre	Description	Valeur par défaut
version		Toujours 1
storageDriverName	Nom du pilote de stockage	ontap-nas, ontap-nas-économie, ontap-nas-flexgroup, ontap-san », « ontap-san », « ontap-économie san »
backendName	Nom personnalisé ou système back-end de stockage	Nom du pilote + "_" + dataLIF
managementLIF	Adresse IP d'un cluster ou d'une LIF de gestion SVM	« 10.0.0.1 », « [2001:1234:abcd::fefe] »
dataLIF	Adresse IP de la LIF de protocole. Utilisez des crochets pour IPv6. Ne peut pas être mis à jour une fois que vous l'avez défini	Dérivé par la SVM sauf spécification
autoExportPolicy	Activer la création et la mise à jour automatiques des règles d'exportation [booléennes]	faux
autoExportCIDRs	Liste des CIDR pour filtrer les adresses IP du nœud Kubernetes par rapport à quand autoExportPolicy est activé	["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	Valeur par 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 CA de confiance. Facultatif. Utilisé pour l'authentification par certificat	« »
username	Nom d'utilisateur pour la connexion au cluster/SVM. Utilisé pour l'authentification basée sur les identifiants	
password	Mot de passe pour la connexion au cluster/SVM. Utilisé pour l'authentification basée sur les identifiants	
svm	Serveur virtuel de stockage à utiliser	Dérivé d'un SVM managementLIF est spécifié
igroupName	Nom du groupe initiateur à utiliser pour les volumes SAN	Trident-<backend-UUID>
storagePrefix	Préfixe utilisé pour le provisionnement des nouveaux volumes dans la SVM. Ne peut pas être mis à jour une fois que vous l'avez défini	trident
limitAggregateUsage	Echec du provisionnement si l'utilisation est supérieure à ce pourcentage. Ne s'applique pas à Amazon FSX pour ONTAP	« » (non appliqué par défaut)
limitVolumeSize	Echec du provisionnement si la taille du volume demandé est supérieure à cette valeur.	« » (non appliqué par défaut)
lunsPerFlexvol	Nombre maximal de LUN par FlexVol, doit être compris dans la plage [50, 200]	"100"
debugTraceFlags	Indicateurs de débogage à utiliser lors du dépannage. Exemple, {"api":false, "méthode":true}	nul
nfsMountOptions	Liste des options de montage NFS séparée par des virgules	« »
qtreesPerFlexvol	Nombre maximal de qtrees par FlexVol, qui doit être compris dans la plage [50, 300]	"200"

Paramètre	Description	Valeur par défaut
useREST	Paramètre booléen pour utiliser les API REST de ONTAP. Aperçu technique	faux



useREST est fourni sous forme d'aperçu technique ** qui est recommandé pour les environnements de test et non pour les charges de travail de production. Lorsqu'il est réglé sur true, Astra Trident va utiliser les API REST de ONTAP pour communiquer avec le système back-end. Cette fonctionnalité requiert ONTAP 9.9 et versions ultérieures. En outre, le rôle de connexion ONTAP utilisé doit avoir accès au ontap client supplémentaire. Ceci est satisfait par le pré-défini vsadmin et cluster-admin rôles.

Pour communiquer avec le cluster ONTAP, vous devez fournir les paramètres d'authentification. Il peut s'agir du nom d'utilisateur/mot de passe d'une connexion de sécurité ou d'un certificat installé.



Si vous utilisez un système Amazon FSX pour le système back-end NetApp ONTAP, ne spécifiez pas le système limitAggregateUsage paramètre. Le fsxadmin et vsadmin Les rôles fournis par Amazon FSX pour NetApp ONTAP ne contiennent pas les autorisations d'accès requises pour récupérer l'utilisation des agrégats et le limiter via Astra Trident.



Ne pas utiliser debugTraceFlags à moins que vous ne soyez en mesure de dépanner et que vous ayez besoin d'un vidage détaillé des journaux.



Lors de la création d'un back-end, n'oubliez pas que le dataLIF et storagePrefix ne peut pas être modifié après sa création. Pour mettre à jour ces paramètres, vous devez créer un nouveau back-end.

Un nom de domaine complet (FQDN) peut être spécifié pour le managementLIF option. Un FQDN peut également être spécifié pour le dataLIF Option, dans ce cas, le FQDN sera utilisé pour les opérations de montage NFS. Cette méthode vous permet de créer un DNS cyclique pour équilibrer la charge entre plusieurs LIF de données.

```
`managementLIF` Pour tous les pilotes ONTAP peuvent également être définis sur des adresses IPv6. Veuillez à installer Astra Trident avec le `--use-ipv6` drapeau. Il faut veiller à définir le `managementLIF` Adresse IPv6 entre crochets.
```



Lorsque vous utilisez des adresses IPv6, assurez-vous de managementLIF et dataLIF (si inclus dans votre définition de back-end) sont définis entre crochets, tels que [28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]. Si dataLIF N'est pas fourni, Astra Trident va récupérer les LIF de données IPv6 à partir du SVM.

À l'aide du autoExportPolicy et autoExportCIDRs CSI Trident peut gérer automatiquement les règles d'exportation. Ceci est pris en charge pour tous les pilotes ontap-nas-*

Pour le ontap-nas-economy conducteur, le limitVolumeSize Elle restreindra également la taille maximale des volumes qu'elle gère pour les qtrees et les LUN, ainsi que qtreesPerFlexvol L'option permet

de personnaliser le nombre maximal de qtrees par FlexVol.

Le `nfsMountOptions` ce paramètre peut être utilisé pour spécifier les options de montage. Les options de montage des 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, Astra Trident utilisera les options de montage spécifiées dans le fichier de configuration du système de stockage back-end. Si aucune option de montage n'est spécifiée dans la classe de stockage ou le fichier de configuration, Astra Trident ne définit aucune option de montage sur un volume persistant associé.



Astra Trident définit les libellés de provisionnement dans le champ « Commentaires » de tous les volumes créés à l'aide de `(ontap-nas)` et `(ontap-nas-flexgroup)`. En fonction du pilote utilisé, les commentaires sont définis sur le FlexVol (`ontap-nas`) Ou FlexGroup (`ontap-nas-flexgroup`). Astra Trident copiera toutes les étiquettes présentes sur un pool de stockage sur le volume de stockage au moment de son provisionnement. Les administrateurs de stockage peuvent définir des étiquettes par pool de stockage et regrouper tous les volumes créés dans un pool de stockage. Cela permet de différencier facilement les volumes en fonction d'un ensemble d'étiquettes personnalisables fournies dans la configuration back-end.

Options de configuration back-end pour les volumes de provisionnement

Vous pouvez contrôler la façon dont chaque volume est provisionné par défaut à l'aide de ces options dans une section spéciale de la configuration. Pour un exemple, voir les exemples de configuration ci-dessous.

Paramètre	Description	Valeur par défaut
<code>spaceAllocation</code>	Allocation d'espace pour les LUN	« vrai »
<code>spaceReserve</code>	Mode de réservation d'espace ; "none" (fin) ou "volume" (épais)	« aucun »
<code>snapshotPolicy</code>	Règle Snapshot à utiliser	« aucun »
<code>qosPolicy</code>	QoS policy group à affecter pour les volumes créés. Choisissez une de <code>qosPolicy</code> ou <code>adaptiveQosPolicy</code> par pool de stockage/back-end	« »
<code>adaptiveQosPolicy</code>	Groupe de règles de QoS adaptative à attribuer aux volumes créés. Choisissez une de <code>qosPolicy</code> ou <code>adaptiveQosPolicy</code> par pool de stockage/back-end. Non pris en charge par l'économie <code>ontap-nas</code> .	« »
<code>snapshotReserve</code>	Pourcentage du volume réservé pour les instantanés "0"	Si <code>snapshotPolicy</code> est « aucun », sinon « »
<code>splitOnClone</code>	Séparer un clone de son parent lors de sa création	« faux »
<code>encryption</code>	Activer le chiffrement de volume NetApp	« faux »
<code>securityStyle</code>	Style de sécurité pour les nouveaux volumes	"unix"

Paramètre	Description	Valeur par défaut
tieringPolicy	La stratégie de hiérarchisation à utiliser « none »	Snapshot uniquement pour une configuration SVM-DR pré-ONTAP 9.5
Autorisations unix	Mode pour les nouveaux volumes	“777”
Dir. Des snapshots	Contrôle la visibilité du .snapshot répertoire	« faux »
ExportPolicy	Export policy à utiliser	« par défaut »
SecurityStyle	Style de sécurité pour les nouveaux volumes	“unix”



Avec Astra Trident, les groupes de règles de QoS doivent être utilisés avec ONTAP 9.8 ou version ultérieure. Il est recommandé d'utiliser un groupe de règles de qualité de service non partagé et de s'assurer que le groupe de règles est appliqué à chaque composant individuellement. Un groupe de règles de QoS partagé appliquera le plafond du débit total de toutes les charges de travail.

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`, Astra Trident utilise maintenant un nouveau calcul pour s'assurer que la FlexVol est correctement dimensionnée avec le pourcentage de snapshots et la demande de volume persistant. Lorsque l'utilisateur demande de volume persistant, Astra Trident crée le FlexVol d'origine

avec plus d'espace en utilisant le nouveau calcul. Ce calcul garantit que l'utilisateur reçoit l'espace inscriptible demandé dans la demande de volume persistant et qu'il ne dispose pas d'un espace minimal par rapport à ce qu'il a demandé. Avant le 21.07, lorsque l'utilisateur demande une demande de volume persistant (par exemple, 5 Gio), et le `snapshotReserve` à 50 %, ils ne bénéficient que d'un espace inscriptible de 2,5 Gio. En effet, le nom d'utilisateur requis correspond à l'intégralité du volume et `snapshotReserve` représente un pourcentage de cela. Avec Trident 21.07, il s'agit de l'espace inscriptible demandé par l'utilisateur et d'Astra Trident définit le `snapshotReserve` nombre comme pourcentage de l'intégralité du volume. Cela ne s'applique pas à `ontap-nas-economy`. Voir l'exemple suivant pour voir comment cela fonctionne :

Le calcul est le suivant :

$$\text{Total volume size} = (\text{PVC requested size}) / (1 - (\text{snapshotReserve percentage}) / 100)$$

Pour les snapshots `Reserve = 50 %`, et demande en volume PVC = 5 Gio, la taille totale du volume est $2/0,5 = 10$ Gio et la taille disponible est de 5 Gio, ce que l'utilisateur a demandé dans la demande de demande de volume persistant. Le `volume show` la commande doit afficher des résultats similaires à cet exemple :

Vserver	Volume	Aggregate	State	Type	Size	Available	Used%
	_pvc_89f1c156_3801_4de4_9f9d_034d54c395f4		online	RW	10GB	5.00GB	0%
	_pvc_e8372153_9ad9_474a_951a_08ae15e1c0ba		online	RW	1GB	511.8MB	0%

2 entries were displayed.

Les systèmes back-end des installations précédentes provisionnent les volumes comme expliqué ci-dessus lors de la mise à niveau d'Astra Trident. Pour les volumes que vous avez créés avant la mise à niveau, vous devez redimensionner leurs volumes afin que la modification puisse être observée. Par exemple, un PVC de 2 Gio avec `snapshotReserve=50` Auparavant, un volume doté d'un espace inscriptible de 1 Gio. Le redimensionnement du volume à 3 Gio, par exemple, fournit l'application avec 3 Gio d'espace inscriptible sur un volume de 6 Gio.

Exemples de configuration minimaux

Les exemples suivants montrent des configurations de base qui laissent la plupart des paramètres par défaut. C'est la façon la plus simple de définir un back-end.



Si vous utilisez Amazon FSX sur NetApp ONTAP avec Trident, nous vous recommandons de spécifier des noms DNS pour les LIF au lieu d'adresses IP.

ontap-nas pilote avec authentification par certificat

Il s'agit d'un exemple de configuration back-end minimal. `clientCertificate`, `clientPrivateKey`, et `trustedCACertificate` (Facultatif, si vous utilisez une autorité de certification approuvée) est renseigné `backend.json` Et prendre les valeurs codées en base64 du certificat client, de la clé privée et du certificat CA de confiance, respectivement.

```

{
  "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_"
}

```

ontap-nas **pilote avec règle d'exportation automatique**

Cet exemple vous montre comment vous pouvez demander à Astra Trident d'utiliser des règles d'exportation dynamiques pour créer et gérer automatiquement la règle d'exportation. Cela fonctionne de la même manière pour le 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": "secret",
  "nfsMountOptions": "nfsvers=4",
}

```

ontap-nas-flexgroup **conducteur**

```

{
  "version": 1,
  "storageDriverName": "ontap-nas-flexgroup",
  "managementLIF": "10.0.0.1",
  "dataLIF": "10.0.0.2",
  "labels": {"k8scluster": "test-cluster-east-1b", "backend": "test1-ontap-cluster"},
  "svm": "svm_nfs",
  "username": "vsadmin",
  "password": "secret",
}

```

ontap-nas **Pilote avec 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": "netapp123"
}

```

ontap-nas-economy **conducteur**

```

{
  "version": 1,
  "storageDriverName": "ontap-nas-economy",
  "managementLIF": "10.0.0.1",
  "dataLIF": "10.0.0.2",
  "svm": "svm_nfs",
  "username": "vsadmin",
  "password": "secret"
}

```

Exemples de systèmes back-end avec pools de stockage virtuel

Dans l'exemple de fichier de définition backend ci-dessous, des valeurs par défaut spécifiques sont définies pour tous les pools de stockage, par exemple `spaceReserve` aucune, `spaceAllocation` lors de la fausse idée, et `encryption` faux. Les pools de stockage virtuels sont définis dans la section `stockage`.

Dans cet exemple, certains pools de stockage sont propriétaires de leur propre pool `spaceReserve`, `spaceAllocation`, et `encryption` les valeurs et certains pools remplacent les valeurs par défaut définies ci-dessus.

ontap-nas **conducteur**

```
{
  {
    "version": 1,
    "storageDriverName": "ontap-nas",
    "managementLIF": "10.0.0.1",
    "dataLIF": "10.0.0.2",
    "svm": "svm_nfs",
    "username": "admin",
    "password": "secret",
    "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":{"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"
    }
  }
]
}

```

ontap-nas-flexgroup **conducteur**

```

{
  "version": 1,
  "storageDriverName": "ontap-nas-flexgroup",
  "managementLIF": "10.0.0.1",
  "dataLIF": "10.0.0.2",
  "svm": "svm_nfs",
  "username": "vsadmin",
  "password": "secret",

  "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"
      }
    }
  ]
}

```

ontap-nas-economy **conducteur**

```

{
  "version": 1,
  "storageDriverName": "ontap-nas-economy",
  "managementLIF": "10.0.0.1",
  "dataLIF": "10.0.0.2",
  "svm": "svm_nfs",
  "username": "vsadmin",
  "password": "secret",

  "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":{"department":"legal", "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"
    }
  }
]
}

```

Mappage des systèmes back-end aux classes de stockage

Les définitions de classe de stockage suivantes font référence aux pools de stockage virtuels ci-dessus. À l'aide du `parameters.selector` Chaque classe de stockage indique quel(s) pool(s) virtuel(s) peut(s) être utilisé(s) pour héberger un volume. Les aspects définis dans le pool virtuel sélectionné seront définis pour le volume.

- La première classe de stockage (`protection-gold`) sera mappé sur le premier, deuxième pool de stockage virtuel dans le `ontap-nas-flexgroup` système back-end et le premier pool de stockage virtuel dans `ontap-san` back-end. Il s'agit du seul pool offrant une protection de niveau Gold.
- La deuxième classe de stockage (`protection-not-gold`) sera mappé sur le troisième, quatrième pool de stockage virtuel dans `ontap-nas-flexgroup` back-end et le deuxième, troisième pool de stockage virtuel dans `ontap-san` back-end. Ce sont les seuls pools offrant un niveau de protection autre que l'or.
- La troisième classe de stockage (`app-mysqldb`) sera mappé sur le quatrième pool de stockage virtuel dans `ontap-nas` back-end et le troisième pool de stockage virtuel dans `ontap-san-economy` back-end. Ce sont les seuls pools offrant une configuration de pool de stockage pour l'application de type `mysqldb`.
- La quatrième classe de stockage (`protection-silver-creditpoints-20k`) sera mappé sur le troisième pool de stockage virtuel dans `ontap-nas-flexgroup` back-end et le second pool de stockage virtuel dans `ontap-san` back-end. Ce sont les seules piscines offrant une protection de niveau or à 20000 points de solvabilité.
- La cinquième classe de stockage (`creditpoints-5k`) sera mappé sur le second pool de stockage virtuel dans `ontap-nas-economy` back-end et le troisième pool de stockage virtuel dans `ontap-san` back-end. Ce sont les seules offres de piscine à 5000 points de solvabilité.

Astra Trident va décider du pool de stockage virtuel sélectionné et s'assurer que les besoins en stockage sont satisfaits.


```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-gold
provisioner: netapp.io/trident
parameters:
  selector: "protection=gold"
  fsType: "ext4"
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-not-gold
provisioner: netapp.io/trident
parameters:
  selector: "protection!=gold"
  fsType: "ext4"
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: app-mysqldb
provisioner: netapp.io/trident
parameters:
  selector: "app=mysqldb"
  fsType: "ext4"
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-silver-creditpoints-20k
provisioner: netapp.io/trident
parameters:
  selector: "protection=silver; creditpoints=20000"
  fsType: "ext4"
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: creditpoints-5k
provisioner: netapp.io/trident
parameters:
  selector: "creditpoints=5000"
  fsType: "ext4"
```

Utilisez Astra Trident avec Amazon FSX pour NetApp ONTAP

"[Amazon FSX pour NetApp ONTAP](#)", Est un service AWS entièrement géré qui permet aux clients de lancer et d'exécuter des systèmes de fichiers optimisés par le système d'exploitation du stockage ONTAP de NetApp. La solution Amazon FSX pour NetApp ONTAP vous permet d'exploiter les fonctionnalités, les performances et les capacités administratives de NetApp que vous connaissez déjà, tout en bénéficiant de la simplicité, de l'agilité, de la sécurité et de l'évolutivité du stockage de données sur AWS. FSX prend en charge de nombreuses fonctionnalités de système de fichiers et API d'administration d'ONTAP.

Un système de fichiers est la ressource principale d'Amazon FSX, similaire à un cluster ONTAP sur site. Au sein de chaque SVM, vous pouvez créer un ou plusieurs volumes, qui sont des conteneurs de données qui stockent les fichiers et les dossiers dans votre système de fichiers. Avec Amazon FSX pour NetApp ONTAP, Data ONTAP sera fourni en tant que système de fichiers géré dans le cloud. Le nouveau type de système de fichiers est appelé **NetApp ONTAP**.

Avec Astra Trident et Amazon FSX pour NetApp ONTAP, vous avez l'assurance que vos clusters Kubernetes s'exécutant dans Amazon Elastic Kubernetes Service (EKS) peuvent provisionner des volumes persistants de bloc et de fichier sauvegardés par ONTAP.

Découvrez Astra Trident

Si vous découvrez Astra Trident, familiarisez-vous avec les liens ci-dessous :

- ["FAQ"](#)
- ["Exigences d'utilisation d'Astra Trident"](#)
- ["Déployez Astra Trident"](#)
- ["Meilleures pratiques de configuration de ONTAP, Cloud Volumes ONTAP et Amazon FSX pour NetApp ONTAP"](#)
- ["Intégrez Astra Trident"](#)
- ["Configuration SAN backend ONTAP"](#)
- ["Configuration NAS backend ONTAP"](#)

En savoir plus sur les capacités des pilotes ["ici"](#).

Utilisation d'Amazon FSX pour NetApp ONTAP ["FabricPool"](#) pour gérer les niveaux de stockage. Elle vous permet de stocker les données au niveau le plus important, selon que celles-ci sont fréquemment utilisées.

Astra Trident devrait être exécuté en tant que `A. vsadmin` Utilisateur SVM ou en tant qu'utilisateur avec un nom différent qui a le même rôle. Amazon FSX pour NetApp ONTAP en a un `fsxadmin` Utilisateur qui remplace le ONTAP de manière limitée `admin` utilisateur du cluster. Il n'est pas recommandé d'utiliser le `fsxadmin` Avec Trident, en tant que `A. vsadmin` Les utilisateurs du SVM ont accès à d'autres fonctionnalités Trident d'Astra,

Pilotes

Vous pouvez intégrer Astra Trident avec Amazon FSX pour NetApp ONTAP à l'aide des pilotes suivants :

- `ontap-san`: Chaque volume persistant provisionné est un LUN au sein de son propre volume Amazon FSX pour NetApp ONTAP.
- `ontap-san-economy`: Chaque volume persistant provisionné est un LUN avec un nombre configurable de LUN par Amazon FSX pour le volume NetApp ONTAP.

- `ontap-nas`: Chaque volume persistant provisionné est un volume Amazon FSX complet pour NetApp ONTAP.
- `ontap-nas-economy`: Chaque volume persistant provisionné est un qtrees, avec un nombre configurable de qtrees par Amazon FSX pour le volume NetApp ONTAP.
- `ontap-nas-flexgroup`: Chaque volume persistant provisionné est un volume Amazon FSX complet pour NetApp ONTAP FlexGroup.

Authentification

Astra Trident propose deux modes d'authentification :

- Basé sur des certificats : Astra Trident communiquera avec le SVM sur votre système de fichiers FSX à l'aide d'un certificat installé sur votre SVM.
- Basé sur les identifiants : vous pouvez utiliser le `fsxadmin` utilisateur pour votre système de fichiers ou `vsadmin` Configuré pour votre SVM.



Nous vous recommandons vivement d'utiliser le `vsadmin` utilisateur au lieu du `fsxadmin` pour configurer votre système backend. Astra Trident communiquera avec le système de fichiers FSX à l'aide de ce nom d'utilisateur et de ce mot de passe.

Pour en savoir plus sur l'authentification, consultez les liens suivants :

- ["NAS ONTAP"](#)
- ["SAN ONTAP"](#)

Déployez et configurez Astra Trident sur EKS avec Amazon FSX pour NetApp ONTAP

Ce dont vous avez besoin

- Un cluster Amazon EKS existant ou un cluster Kubernetes autogéré avec `kubectl` installé.
- Un système de fichiers Amazon FSX pour NetApp ONTAP existant et une machine virtuelle de stockage (SVM) accessible depuis les nœuds workers de votre cluster.
- Nœuds worker prêts pour ["NFS et/ou iSCSI"](#).



Assurez-vous de suivre les étapes de préparation des nœuds requises pour Amazon Linux et Ubuntu ["Images de machine Amazon"](#) (AMIS) en fonction de votre type ami EKS.

Pour connaître les autres exigences d'Astra Trident, consultez le site ["ici"](#).

Étapes

1. Déployez Astra Trident avec l'un des [../trident-get-Started/kubernetes-Deploy.html](#)[méthodes de déploiement^].
2. Configurez Astra Trident comme suit :
 - a. Collectez le nom DNS de la LIF de gestion de votre SVM. Par exemple, en utilisant l'interface de ligne de commandes AWS, recherchez le `DNSName` entrée sous `Endpoints` → `Management` après avoir exécuté la commande suivante :

```
aws fsx describe-storage-virtual-machines --region <file system
region>
```

3. Créer et installer des certificats pour l'authentification. Si vous utilisez un `ontap-san` back-end, voir ["ici"](#). Si vous utilisez un `ontap-nas` back-end, voir ["ici"](#).



Vous pouvez vous connecter à votre système de fichiers (par exemple pour installer des certificats) à l'aide de SSH à partir de n'importe quel endroit qui peut atteindre votre système de fichiers. Utilisez le `fsxadmin` User, le mot de passe que vous avez configuré lors de la création de votre système de fichiers et le nom DNS de gestion à partir de `aws fsx describe-file-systems`.

4. Créer un fichier backend en utilisant vos certificats et le nom DNS de votre LIF de gestion, comme indiqué dans l'exemple ci-dessous :

```
{
  "version": 1,
  "storageDriverName": "ontap-san",
  "backendName": "customBackendName",
  "managementLIF": "svm-XXXXXXXXXXXXXXXXXXXX.fs-XXXXXXXXXXXXXXXXXXXX.fsx.us-
east-2.aws.internal",
  "svm": "svm01",
  "clientCertificate": "ZXR0ZXJwYXB...ICMgJ3BhcGVyc2",
  "clientPrivateKey": "vcIwKIyAgZG...0cnksIGRlc2NyaX",
  "trustedCACertificate": "zcyBbaG...b3Igb3duIGNsYXNz",
}
```

Pour plus d'informations sur la création des systèmes back-end, voir les liens suivants :

- ["Configurer un système back-end avec les pilotes NAS ONTAP"](#)
- ["Configurer un système back-end avec les pilotes SAN ONTAP"](#)



Ne pas spécifier `dataLIF` pour le `ontap-san` et `ontap-san-economy` Pilotes permettant à Astra Trident d'utiliser le chemin d'accès multivoie.



Avec Amazon FSX pour NetApp ONTAP avec Astra Trident, le `limitAggregateUsage` le paramètre ne fonctionne pas avec le `vsadmin` et `fsxadmin` comptes d'utilisateur. L'opération de configuration échoue si vous spécifiez ce paramètre.

Après le déploiement, suivez les étapes pour créer un ["classe de stockage, provisionnez un volume et montez le volume dans un pod"](#).

Trouvez plus d'informations

- ["Documentation Amazon FSX pour NetApp ONTAP"](#)

- ["Billet de blog sur Amazon FSX pour NetApp ONTAP"](#)

Création de systèmes back-end avec kubectl

Un système back-end définit la relation entre Astra Trident et un système de stockage. Il explique à Astra Trident comment communiquer avec ce système de stockage et comment Astra Trident doit provisionner des volumes à partir de celui-ci. Après l'installation d'Astra Trident, l'étape suivante consiste à créer un système back-end. Le `TridentBackendConfig` La définition de ressource personnalisée (CRD) vous permet de créer et de gérer des systèmes back-end 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 un CRD au rythme de noms qui vous permet de gérer les systèmes back-end Astra Trident à l'aide de `kubectl`. Avec Kubernetes et les administrateurs de stockage, il est désormais possible de créer et de gérer des systèmes back-end directement via la CLI Kubernetes sans avoir besoin d'un utilitaire de ligne de commande dédié (`tridentctl`).

Lors de la création d'un `TridentBackendConfig` objet :

- Un système back-end est créé automatiquement par Astra Trident en fonction de la configuration que vous fournissez. Il est représenté en interne en tant que `TridentBackend` (`tbe`, `tridentbackend`) CR.
- Le `TridentBackendConfig` est lié de manière unique à un `TridentBackend` Créé par Astra Trident.

Chacun `TridentBackendConfig` gère un mappage un-à-un avec un `TridentBackend`. La première est l'interface fournie à l'utilisateur pour concevoir et configurer les systèmes back-end, tandis que `Trident` représente l'objet back-end réel.



`TridentBackend` ASTRA Trident crée automatiquement des CRS. Vous ne devez pas les modifier. Si vous voulez effectuer des mises à jour vers les systèmes back-end, modifiez le `TridentBackendConfig` objet.

Reportez-vous à l'exemple suivant pour connaître 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 de la "[programme d'installation trident](#)" répertoire des exemples de configuration pour la plate-forme/le service de stockage souhaité.

Le `spec` il prend des paramètres de configuration spécifiques au back-end. Dans cet exemple, le back-end utilise le `ontap-san` pilote de stockage et utilise les paramètres de configuration qui sont présentés ici. Pour obtenir la liste des options de configuration pour le pilote de stockage souhaité, reportez-vous à la section "[informations de configuration backend pour votre pilote de stockage](#)".

Le `spec` la section inclut également `credentials` et `deletionPolicy` les champs qui viennent d'être introduits dans le `TridentBackendConfig` CR :

- `credentials`: Ce paramètre est un champ obligatoire et contient les informations d'identification utilisées pour s'authentifier auprès du système/service de stockage. Cette configuration est définie sur un code secret Kubernetes créé par l'utilisateur. Les informations d'identification ne peuvent pas être transmises en texte brut et entraînent une erreur.
- `deletionPolicy`: Ce champ définit ce qui doit se produire lorsque `TridentBackendConfig` est supprimé. Il peut prendre l'une des deux valeurs possibles :
 - `delete`: Cela entraîne la suppression des deux `TridentBackendConfig` CR et le back-end associé. Il s'agit de la valeur par défaut.
 - `retain`: Lorsqu'un `TridentBackendConfig` La demande de modification est supprimée, la définition de l'arrière-plan est toujours présente et peut être gérée avec `tridentctl`. Définition de la stratégie de suppression sur `retain` permet aux utilisateurs de revenir à une version antérieure (avant la version 21.04) et de conserver les systèmes back-end créés. La valeur de ce champ peut être mise à jour après un `TridentBackendConfig` est créé.



Le nom d'un backend est défini à l'aide de `spec.backendName`. S'il n'est pas spécifié, le nom du back-end est défini sur le nom du `TridentBackendConfig` objet (`metadata.name`). Il est recommandé de définir explicitement les noms backend à l'aide de `spec.backendName`.



Systèmes back-end créés avec `tridentctl` n'avez pas de lien associé `TridentBackendConfig` objet. Vous avez la possibilité de choisir de gérer de tels systèmes back-end avec `kubectl` en créant un `TridentBackendConfig` CR. Vous devez veiller à spécifier des paramètres de configuration identiques (par exemple `spec.backendName`, `spec.storagePrefix`, `spec.storageDriverName`, etc.). Astra Trident lie automatiquement le nouveau produit `TridentBackendConfig` avec le back-end existant.

Présentation des étapes

Pour créer un nouveau back-end à l'aide de `kubectl`, vous devez effectuer les opérations suivantes :

1. Créer un "Le secret de Kubernetes". Le secret est qu'Astra Trident doit communiquer avec le cluster/service de stockage.
2. Créer un `TridentBackendConfig` objet. Elle contient des informations spécifiques sur le cluster/service de stockage et fait référence au secret créé à l'étape précédente.

Après avoir créé un back-end, vous pouvez observer son état en utilisant `kubectl get tbc <tbc-name> -n <trident-namespace>` et recueillez des détails supplémentaires.

Étape 1 : créez un code secret Kubernetes

Créez un secret qui contient les informations d'identification d'accès pour le back-end. Ce point 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: t@Ax@7q(>
```

Ce tableau récapitule les champs à inclure dans le Secret pour chaque plate-forme de stockage :

Description des champs secrets de la plate-forme de stockage	Secret	Description des champs
Azure NetApp Files	ID client	ID client d'un enregistrement d'application
Cloud Volumes Service pour AWS	ApiKey	Clé API de compte CVS
Cloud Volumes Service pour AWS	SecretKey	Clé secrète compte CVS

Description des champs secrets de la plate-forme de stockage	Secret	Description des champs
Cloud Volumes Service pour GCP	id_clé_privée	ID de la clé privée. Partie de la clé API pour le compte de service GCP avec le rôle d'administrateur CVS
Cloud Volumes Service pour GCP	clé_privée	Clé privée. Partie de la clé API pour le compte de service GCP avec le rôle d'administrateur CVS
Element (NetApp HCI/SolidFire)	Point final	MVIP pour le cluster SolidFire avec les identifiants de locataire
ONTAP	nom d'utilisateur	Nom d'utilisateur pour la connexion au cluster/SVM. Utilisé pour l'authentification basée sur les identifiants
ONTAP	mot de passe	Mot de passe pour la connexion au cluster/SVM. Utilisé pour l'authentification basée sur les identifiants
ONTAP	ClientPrivateKey	Valeur encodée en Base64 de la clé privée du client. Utilisé pour l'authentification basée sur des certificats
ONTAP	ChapUsername	Nom d'utilisateur entrant. Requis si useCHAP=vrai. Pour <code>ontap-san</code> et <code>ontap-san-economy</code>
ONTAP	Chapeau InitiatorSecret	Secret de l'initiateur CHAP. Requis si useCHAP=vrai. Pour <code>ontap-san</code> et <code>ontap-san-economy</code>
ONTAP	ChapTargetUsername	Nom d'utilisateur cible. Requis si useCHAP=vrai. Pour <code>ontap-san</code> et <code>ontap-san-economy</code>
ONTAP	ChapTargetInitiatorSecret	Secret de l'initiateur cible CHAP. Requis si useCHAP=vrai. Pour <code>ontap-san</code> et <code>ontap-san-economy</code>

Le secret créé dans cette étape sera référencé dans le `spec.credentials` champ du `TridentBackendConfig` objet créé à l'étape suivante.

Étape 2 : créez le TridentBackendConfig CR

Vous êtes maintenant prêt à créer votre TridentBackendConfig CR. Dans cet exemple, un back-end qui utilise le ontap-san le pilote est créé à l'aide du TridentBackendConfig objet illustré 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 : vérifier l'état 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 back-end a été créé avec succès et lié au TridentBackendConfig CR.

La phase peut prendre l'une des valeurs suivantes :

- **Bound:** Le TridentBackendConfig La demande de modification est associée à un back-end, et ce backend contient configRef réglez sur TridentBackendConfig L'uid de CR.
- **Unbound:** Représenté en utilisant "". Le TridentBackendConfig l'objet n'est pas lié à un back-end. Tout nouveau TridentBackendConfig Les CRS sont dans cette phase par défaut. Une fois la phase modifiée, elle ne peut plus revenir à Unbound.
- **Deleting:** Le TridentBackendConfig CR deletionPolicy a été configuré pour supprimer. Lorsque le TridentBackendConfig La demande de modification est supprimée, elle passe à l'état Suppression.

- Si aucune demande de volume persistant n'existe sur le back-end, supprimez le `TridentBackendConfig` Il en résultera la suppression du système back-end et du système Astra `TridentBackendConfig` CR.
- Si un ou plusieurs ESV sont présents sur le back-end, il passe à l'état de suppression. Le `TridentBackendConfig` La CR entre ensuite la phase de suppression. Le back-end et `TridentBackendConfig` Sont supprimés uniquement après la suppression de tous les ESV.
- **Lost:** Le back-end associé à l' `TridentBackendConfig` Le CR a été accidentellement ou délibérément supprimé et le `TridentBackendConfig` La CR a toujours une référence au back-end supprimé. Le `TridentBackendConfig` La CR peut toujours être supprimée, quel que soit le `deletionPolicy` valeur.
- **Unknown:** Astra Trident n'est pas en mesure de déterminer l'état ou l'existence du back-end associé au `TridentBackendConfig` CR. Par exemple, si le serveur d'API ne répond pas ou si `tridentbackends.trident.netapp.io` CRD manquant. Cela peut nécessiter l'intervention de l'utilisateur.

À ce stade, un système back-end est créé avec succès ! Plusieurs opérations peuvent également être traitées, par exemple "[mises à jour du système back-end et suppressions](#)".

(Facultatif) étape 4 : pour plus de détails

Vous pouvez exécuter la commande suivante pour obtenir plus d'informations sur votre système back-end :

```
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

En outre, vous pouvez également obtenir un vidage 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 le backendName et le backendUUID du back-end créé en réponse à TridentBackendConfig CR. Le lastOperationStatus champ représente l'état de la dernière opération du TridentBackendConfig CR, qui peut être déclenché par l'utilisateur (par exemple, l'utilisateur a modifié quelque chose dans spec) Ou déclenché par Astra Trident (par exemple lors du redémarrage d'Astra Trident). Il peut être réussi ou échoué. phase représente l'état de la relation entre TridentBackendConfig CR et le backend. Dans l'exemple ci-dessus, phase a la valeur limitée, ce qui signifie que le TridentBackendConfig CR est associé au back-end.

Vous pouvez exécuter le `kubectl -n trident describe tbc <tbc-cr-name>` commande pour obtenir des détails sur les journaux d'événements.



Vous ne pouvez pas mettre à jour ou supprimer un backend qui contient un associé TridentBackendConfig objet utilisant tridentctl. Pour comprendre les étapes de passage d'un à l'autre tridentctl et TridentBackendConfig, ["voir ici"](#).

Effectuer la gestion back-end avec kubectl

Découvrez comment effectuer des opérations de gestion back-end à l'aide de `kubectl`.

Supprimer un back-end

En supprimant un `TridentBackendConfig`, Vous demandez à Astra Trident de supprimer/conservé les systèmes back-end (sur la base `deletionPolicy`). Pour supprimer un back-end, assurez-vous que `deletionPolicy` est configuré pour supprimer. Pour supprimer uniquement le `TridentBackendConfig`, assurez-vous que `deletionPolicy` est défini sur conserver. Cela permet de s'assurer que le système backend est toujours présent et qu'il peut être géré à l'aide de `tridentctl`.

Exécutez la commande suivante :

```
$ kubectl delete tbc <tbc-name> -n trident
```

Astra Trident ne supprime pas les secrets Kubernetes qui étaient utilisés par `TridentBackendConfig`. L'utilisateur Kubernetes est chargé de nettoyer les secrets. Il faut faire attention lors de la suppression des secrets. Vous devez supprimer les secrets uniquement s'ils ne sont pas utilisés par les systèmes back-end.

Affichez les systèmes back-end 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 systèmes back-end existants, Cette liste comprend également les systèmes back-end créés avec `tridentctl`.

Mettre à jour un back-end

Il peut y avoir plusieurs raisons de mettre à jour un backend :

- Les informations d'identification du système de stockage ont été modifiées. Pour mettre à jour les identifiants, le code secret Kubernetes utilisé dans le `TridentBackendConfig` l'objet doit être mis à jour. Avec Astra Trident, le système back-end est automatiquement mis à jour avec les dernières informations d'identification fournies. Exécutez la commande suivante pour mettre à jour le code secret Kubernetes :

```
$ kubectl apply -f <updated-secret-file.yaml> -n trident
```

- Les paramètres (tels que le nom du SVM ONTAP utilisé) doivent être mis à jour. Dans ce cas, `TridentBackendConfig` Les objets peuvent être mis à jour directement via Kubernetes.

```
$ kubectl apply -f <updated-backend-file.yaml>
```

Vous pouvez également apporter des modifications à l'existant `TridentBackendConfig` CR en exécutant la commande suivante :

```
$ kubectl edit tbc <tbc-name> -n trident
```

En cas d'échec d'une mise à jour du back-end, le système back-end continue de rester dans sa dernière configuration connue. Vous pouvez afficher les journaux pour déterminer la cause en cours d'exécution `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 `update`.

Gestion back-end avec `tridentctl`

Découvrez comment effectuer des opérations de gestion back-end à l'aide de `tridentctl`.

Créer un back-end

Après avoir créé un "[fichier de configuration back-end](#)", exécutez la commande suivante :

```
$ tridentctl create backend -f <backend-file> -n trident
```

Si la création du système back-end échoue, la configuration du système back-end était erronée. Vous pouvez afficher les journaux pour déterminer la cause en exécutant la commande suivante :

```
$ tridentctl logs -n trident
```

Une fois que vous avez identifié et corrigé le problème avec le fichier de configuration, vous pouvez simplement exécuter le `create` commande de nouveau.

Supprimer un back-end

Pour supprimer un back-end d'Astra Trident, procédez comme suit :

1. Récupérer le nom du système back-end :

```
$ tridentctl get backend -n trident
```

2. Supprimer le backend :

```
$ tridentctl delete backend <backend-name> -n trident
```



Si Astra Trident a provisionné des volumes et des snapshots à partir de ce backend qui existe toujours, la suppression du back-end empêche les nouveaux volumes d'être provisionnés. Le système back-end continuera à exister dans un état « Suppression » et Trident continuera à gérer ces volumes et ces snapshots jusqu'à leur suppression.

Affichez les systèmes back-end existants

Pour afficher les systèmes back-end dont Trident a conscience, procédez comme suit :

- Pour obtenir un récapitulatif, 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 back-end

Après avoir créé un nouveau fichier de configuration back-end, exécutez la commande suivante :

```
$ tridentctl update backend <backend-name> -f <backend-file> -n trident
```

En cas d'échec de la mise à jour back-end, quelque chose était incorrect avec la configuration back-end ou vous avez tenté une mise à jour non valide. Vous pouvez afficher les journaux pour déterminer la cause en exécutant la commande suivante :

```
$ tridentctl logs -n trident
```

Une fois que vous avez identifié et corrigé le problème avec le fichier de configuration, vous pouvez simplement exécuter le `update` commande de nouveau.

Identifier les classes de stockage qui utilisent un système back-end

Voici un exemple de questions que vous pouvez répondre avec le fichier JSON `tridentctl` sorties des objets back-end. Ceci utilise le `jq` utilitaire que vous devez installer.

```
$ tridentctl get backend -o json | jq '[.items[] | {backend: .name, storageClasses: [.storage[].storageClasses]|unique}]'
```

Cela s'applique également aux systèmes back-end créés par l'utilisation `TridentBackendConfig`.

Passez d'une option de gestion back-end à une autre

Découvrez les différentes façons de gérer les systèmes back-end avec Astra Trident. Avec l'introduction de `TridentBackendConfig`, les administrateurs ont désormais deux méthodes uniques de gestion des systèmes back-end. Ceci pose les questions suivantes :

- Les systèmes back-end peuvent être créés avec `tridentctl` être géré avec `TridentBackendConfig`?
- Les systèmes back-end peuvent être créés avec `TridentBackendConfig` gestion via `tridentctl`?

Gérez `tridentctl` utilisation de systèmes back-end `TridentBackendConfig`

Cette section aborde les étapes requises pour gérer les systèmes back-end créés à l'aide de `tridentctl` Directement via l'interface Kubernetes en créant la `TridentBackendConfig` objets.

Cela s'applique aux scénarios suivants :

- Les systèmes back-end préexistants, qui n'ont pas de système `TridentBackendConfig` parce qu'ils ont été créés avec `tridentctl`.
- Nouveaux systèmes back-end créés avec `tridentctl`, tandis que d'autres `TridentBackendConfig` les objets existent.

Dans les deux cas, le système back-end restera présent. Avec Astra Trident, qui planifie les volumes et les exécute. Les administrateurs peuvent choisir l'une des deux options suivantes :

- Continuer à utiliser `tridentctl` pour gérer les systèmes back-end créés en utilisant ces systèmes.
- Lier les systèmes back-end créés à l'aide de `tridentctl` à un nouveau `TridentBackendConfig` objet. Ainsi, le système back-end sera géré à l'aide de `kubectl` et non `tridentctl`.

Pour gérer un système back-end existant à l'aide de `kubectl`, vous devez créer un `TridentBackendConfig` cela se lie au back-end existant. Voici un aperçu du fonctionnement de ces éléments :

1. Créez un code secret Kubernetes. Le secret est qu'Astra Trident doit communiquer avec le cluster/service de stockage.
2. Créer un `TridentBackendConfig` objet. Elle contient des informations spécifiques sur le cluster/service de stockage et fait référence au secret créé à l'étape précédente. Vous devez veiller à spécifier des paramètres de configuration identiques (par exemple `spec.backendName`, `spec.storagePrefix`, `spec.storageDriverName`, etc.). `spec.backendName` doit être défini sur le nom du back-end existant.

Étape 0 : identifier le back-end

Pour créer un `TridentBackendConfig` qui se lie à un back-end existant, vous devrez obtenir la configuration du backend. Dans cet exemple, supposons qu'un back-end a 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 : créez un code secret Kubernetes

Créez un secret qui contient les informations d'identification du back-end, 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 lie automatiquement au pré-existant `ontap-nas-backend` (comme dans cet exemple). Assurez-vous que les exigences suivantes sont respectées :

- Le même nom de back-end est défini dans `spec.backendName`.
- Les paramètres de configuration sont identiques au back-end d'origine.
- Les pools de stockage virtuel (le cas échéant) doivent conserver le même ordre que dans le back-end d'origine.
- Les identifiants sont fournis via un code secret Kubernetes et non en texte brut.

Dans ce cas, le `TridentBackendConfig` se présente comme suit :

```

$ 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 : vérifier l'état du TridentBackendConfig CR

Après le TridentBackendConfig a été créée, sa phase doit être Bound. Il devrait également refléter le même nom de back-end et UUID que celui du back-end existant.

```
$ kubectl -n trident 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 système back-end sera désormais entièrement géré à l'aide du système tbc-ontap-nas-backend TridentBackendConfig objet.

Gérez TridentBackendConfig utilisation de systèmes back-end tridentctl

`tridentctl` possibilité d'afficher la liste des systèmes back-end créés à l'aide de `TridentBackendConfig`. En outre, les administrateurs ont la possibilité de choisir entre la gestion complète de ces systèmes back-end `tridentctl` en supprimant `TridentBackendConfig` et en fait bien sûr `spec.deletionPolicy` est défini sur `retain`.

Étape 0 : identifier le back-end

Par exemple, supposons que le back-end suivant a 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 |
+-----+-----+-----+-----+
+-----+-----+

```

À partir de la sortie, on voit cela TridentBackendConfig A été créé avec succès et est lié à un back-end [observez l'UUID du backend].

Étape 1 : confirmer deletionPolicy est défini sur retain

Passons en revue les avantages de deletionPolicy. Il doit être défini sur retain. Cela permet de s'assurer que lorsqu'un TridentBackendConfig La demande de modification est supprimée, la définition de l'arrière-plan est toujours présente et peut 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

```



Ne pas passer à l'étape suivante sauf si `deletionPolicy` est défini sur `retain`.

Étape 2 : supprimez le `TridentBackendConfig` CR

La dernière étape consiste à supprimer le `TridentBackendConfig` CR. Après avoir confirmé le `deletionPolicy` est défini sur `retain`, vous pouvez poursuivre 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 du `TridentBackendConfig` Objet : Astra Trident la supprime simplement sans le système back-end.

Gérer les classes de stockage

Recherchez des informations sur la création d'une classe de stockage, la suppression d'une classe de stockage et l'affichage des classes de stockage existantes.

Concevez une classe de stockage

Voir ["ici"](#) pour plus d'informations sur les classes de stockage et leur configuration.

Créer une classe de stockage

Après avoir un fichier de classe de stockage, exécutez la commande suivante :

```
kubectl create -f <storage-class-file>
```

`<storage-class-file>` doit être remplacé par votre nom de fichier de classe de stockage.

Supprimer une classe de stockage

Pour supprimer une classe de stockage de Kubernetes, exécutez la commande suivante :

```
kubectl delete storageclass <storage-class>
```

<storage-class> doit être remplacé par votre classe de stockage.

Tout volume persistant créé dans le cadre de cette classe de stockage n'est pas affecté. Astra Trident va continuer à les gérer.



L'ASTRA Trident applique un blanc `fsType` pour les volumes qu'elle crée. Pour les systèmes back-end iSCSI, il est recommandé d'appliquer la configuration `parameters.fsType` Dans la classe de stockage. Vous devez supprimer les classes de stockage existantes et les recréer à l'aide de `parameters.fsType` spécifié.

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 de la classe de stockage Kubernetes, exécutez la commande suivante :

```
kubectl get storageclass <storage-class> -o json
```

- Pour afficher les classes de stockage synchronisées d'Astra Trident, exécutez la commande suivante :

```
tridentctl get storageclass
```

- Pour afficher les détails de la classe de stockage synchronisée d'Astra Trident, exécutez la commande suivante :

```
tridentctl get storageclass <storage-class> -o json
```

Définir une classe de stockage par défaut

Kubernetes 1.6 a ajouté la possibilité de définir une classe de stockage par défaut. Cette classe de stockage sera utilisée pour provisionner un volume persistant si un utilisateur ne en spécifie pas une dans une demande de volume persistant.

- Définissez une classe de stockage par défaut en définissant l'annotation `storageclass.kubernetes.io/is-default-class` vrai dans la définition de classe de stockage. Selon la spécification, toute autre valeur ou absence de l'annotation est interprétée comme fausse.
- 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 à l'aide de 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 du programme d'installation de Trident qui incluent cette annotation.



Vous ne devez avoir qu'une seule classe de stockage par défaut dans votre cluster à un moment donné. Kubernetes n'empêche pas techniquement d'en avoir plusieurs, mais il se comporte comme s'il n'existait aucune classe de stockage par défaut.

Identifier le système back-end pour une classe de stockage

Voici un exemple de questions que vous pouvez répondre avec le fichier JSON `tridentctl` Sorties pour les objets back-end Astra Trident. Ceci utilise le `jq` utilitaire, que vous devrez peut-être installer en premier.

```
tridentctl get storageclass -o json | jq '[.items[] | {storageClass:  
.Config.name, backends: [.storage]|unique}]'
```

Réaliser des opérations de volume

Découvrez les fonctionnalités d'Astra Trident pour la gestion de vos volumes.

- ["Utiliser la topologie CSI"](#)
- ["Travailler avec des instantanés"](#)
- ["Développement des volumes"](#)
- ["Importer des volumes"](#)

Utiliser la topologie CSI

Astra Trident peut créer et relier de façon sélective des volumes aux nœuds présents dans un cluster Kubernetes en utilisant le ["Fonction de topologie CSI"](#). Grâce à la fonction de topologie CSI, l'accès aux volumes peut être limité à un sous-ensemble de nœuds, en fonction des régions et des zones de disponibilité. Les fournisseurs cloud permettent aujourd'hui aux administrateurs Kubernetes de frayer des nœuds basés sur une zone. Les nœuds peuvent être situés dans différentes régions d'une zone de disponibilité ou sur plusieurs zones de disponibilité. Astra Trident utilise la topologie CSI pour faciliter le provisionnement des volumes pour les charges de travail dans une architecture multi-zones.



En savoir plus sur la fonction de topologie CSI ["ici"](#).

Kubernetes propose deux modes de liaison de volumes :

- Avec `VolumeBindingMode` réglé sur `Immediate`, Astra Trident crée le volume sans la reconnaissance de la topologie. La liaison de volumes et le provisionnement dynamique sont gérés au moment de la création de la demande de volume persistant. Il s'agit de la valeur par défaut `VolumeBindingMode` et convient aux clusters qui n'appliquent pas les contraintes de topologie. Les volumes persistants sont créés sans dépendance vis-à-vis des exigences de planification du pod qui en fait la demande.
- Avec `VolumeBindingMode` réglé sur `WaitForFirstConsumer`, La création et la liaison d'un volume persistant pour une demande de volume persistant sont retardées jusqu'à ce qu'un pod qui utilise la demande de volume persistant soit planifié et créé. De cette façon, les volumes sont créés pour répondre aux contraintes de planification appliquées en fonction des besoins de topologie.



Le `WaitForFirstConsumer` le mode de liaison ne nécessite pas d'étiquettes de topologie. Il peut être utilisé indépendamment de la fonction de topologie CSI.

Ce dont vous avez besoin

Pour utiliser la topologie CSI, vous devez disposer des éléments suivants :

- Cluster Kubernetes exécutant la version 1.17 ou ultérieure.

```
$ 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 avoir des étiquettes qui permettent la prise en charge de la topologie (`topology.kubernetes.io/region` et `topology.kubernetes.io/zone`). Ces étiquettes **doivent être présentes sur les nœuds du cluster** avant d'installer Astra Trident pour qu'Astra Trident soit compatible avec 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","kuber-
netes.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","kuber-
netes.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","kuber-
netes.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 : création d'un back-end conscient de la topologie

Les systèmes back-end de stockage Astra Trident peuvent être conçus pour provisionner des volumes de manière sélective selon les zones de disponibilité. Chaque système back-end peut être équipé d'une option `supportedTopologies` bloc qui représente une liste de zones et de régions qui doivent être prises en charge. Pour les classes de stockage qui utilisent un tel backend, un volume ne sera créé que si une application est planifiée dans une région/zone prise en charge.

Voici à quoi ressemble une définition de back-end :

```
{
  "version": 1,
  "storageDriverName": "ontap-san",
  "backendName": "san-backend-us-east1",
  "managementLIF": "192.168.27.5",
  "svm": "iscsi_svm",
  "username": "admin",
  "password": "xxxxxxxxxxxx",
  "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"}
  ]
}
```



`supportedTopologies` sert à fournir une liste de régions et de zones par backend. Ces régions et ces zones représentent la liste des valeurs admissibles qui peuvent être fournies dans une classe de stockage. Pour les classes de stockage qui contiennent un sous-ensemble de régions et de zones qu'il fournit en back-end, Astra Trident crée un volume en interne.

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": "Netapp123",
"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"},
    "region": "Iowa-DC",
    "zone": "Iowa-DC-A",
    "supportedTopologies": [
      {"topology.kubernetes.io/region": "us-central1",
"topology.kubernetes.io/zone": "us-central1-a"}
    ]
  },
  {
    "labels": {"workload":"dev"},
    "region": "Iowa-DC",
    "zone": "Iowa-DC-B",
    "supportedTopologies": [
      {"topology.kubernetes.io/region": "us-central1",
"topology.kubernetes.io/zone": "us-central1-b"}
    ]
  }
]
}

```

Dans cet exemple, le `region` et `zone` les étiquettes correspondent à l'emplacement du pool de stockage. `topology.kubernetes.io/region` et `topology.kubernetes.io/zone` déterminer à partir de où les pools de stockage peuvent être consommés.

Étape 2 : définissez des classes de stockage qui prennent en compte la topologie

Les classes de stockage peuvent être définies en fonction des labels de topologie fournis aux nœuds du cluster, et contenir des informations de topologie. Cela déterminera les pools de stockage qui servent de candidats aux demandes de volume persistant faites et le sous-ensemble de nœuds qui peuvent 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 de classe de stockage décrite ci-dessus, `volumeBindingMode` est défini sur `WaitForFirstConsumer`. Les demandes de volume persistant demandées pour cette classe de stockage ne seront pas traitées tant qu'elles ne seront pas référencées dans un pod. Et, `allowedTopologies` fournit les zones et la région à utiliser. Le `netapp-san-us-east1` `StorageClass` crée des ESV sur le `san-backend-us-east1` système back-end défini ci-dessus.

Étape 3 : création et utilisation d'une demande de volume persistant

Une fois la classe de stockage créée et mappée à un back-end, vous pouvez désormais créer des demandes de volume persistant.

Voir l'exemple `spec` ci-dessous :

```

---
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
name: pvc-san
spec:
accessModes:
  - ReadWriteOnce
resources:
  requests:
    storage: 300Mi
storageClassName: netapp-san-us-east1

```

La création d'une demande de volume persistant à l'aide de ce manifeste se traduit par les éléments suivants :

```

$ kubectl create -f pvc.yaml
persistentvolumeclaim/pvc-san created
$ kubectl get pvc
NAME          STATUS      VOLUME      CAPACITY   ACCESS MODES   STORAGECLASS
AGE
pvc-san      Pending
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
  ----      -
  Normal    WaitForFirstConsumer 6s    persistentvolume-controller
waiting for first consumer to be created before binding
  Message
  -----

```

Pour que Trident puisse créer un volume et le lier à la demande de volume persistant, utilisez la demande de volume persistant 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: voll
      persistentVolumeClaim:
        claimName: pvc-san
  containers:
    - name: sec-ctx-demo
      image: busybox
      command: [ "sh", "-c", "sleep 1h" ]
      volumeMounts:
        - name: voll
          mountPath: /data/demo
      securityContext:
        allowPrivilegeEscalation: false

```

Ce podSpec demande à Kubernetes de planifier le pod sur les nœuds présents dans le us-east1 et choisissez parmi les nœuds présents dans le us-east1-a ou us-east1-b zones.

Voir le résultat suivant :

```

$ 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

```

Mise à jour des systèmes back-end pour inclure `supportedTopologies`

Les systèmes back-end pré-existants peuvent être mis à jour pour inclure une liste de `supportedTopologies` à l'aide de `tridentctl backend update`. Cela n'affecte pas les volumes qui ont déjà été provisionnés et ne sera utilisé que pour les demandes de volume virtuel suivantes.

Trouvez plus d'informations

- ["Gestion des ressources pour les conteneurs"](#)
- ["Outil de sélection de nœud"](#)
- ["Affinité et anti-affinité"](#)
- ["Teintes et tolérances"](#)

Travailler avec des instantanés

À partir de la version 20.01 d'Astra Trident, vous pouvez créer des snapshots de volumes persistants à la couche Kubernetes. Vous pouvez utiliser ces snapshots pour conserver des copies instantanées de volumes créés par Astra Trident et planifier la création de volumes supplémentaires (clones). Le snapshot de volume est pris en charge par `ontap-nas`, `ontap-nas-flexgroup`, `ontap-san`, `ontap-san-economy`, `solidfire-san`, `aws-cvs`, `gcp-cvs`, et `azure-netapp-files` pilotes.



Cette fonctionnalité est disponible à partir de Kubernetes 1.17 (version bêta) et est disponible dans la version 1.20. Pour comprendre les changements impliqués dans le passage de la version bêta à GA, voir ["le blog de release"](#). Avec la graduation à GA, le `v1` La version d'API est introduite et rétrocompatible avec `v1beta1` snapshots.

Ce dont vous avez besoin

- La création de snapshots de volumes nécessite la création d'un contrôleur d'instantanés externe ainsi que de certaines définitions de ressources personnalisées (CRD). Il s'agit de la responsabilité de l'orchestrateur Kubernetes utilisé (par exemple : Kubeadm, GKE, OpenShift).

Vous pouvez créer un contrôleur de snapshot externe et des CRD de snapshot comme suit :

1. Création de CRD de snapshot de volume :

```
$ cat snapshot-setup.sh
#!/bin/bash
# Create volume snapshot CRDs
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
3.0/client/config/crd/snapshot.storage.k8s.io_volumesnapshotclasses.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
3.0/client/config/crd/snapshot.storage.k8s.io_volumesnapshotcontents.yam
l
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
3.0/client/config/crd/snapshot.storage.k8s.io_volumesnapshots.yaml
```

2. Créer le snapshot-contrôleur dans l'espace de noms souhaité. Modifiez les manifestes YAML ci-dessous pour modifier l'espace de noms.



Ne créez pas de snapshot-contrôleur si vous configurez des snapshots de volume à la demande dans un environnement GKE. GKE utilise un snapshot-contrôleur intégré et masqué.

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-3.0/deploy/kubernetes/snapshot-
controller/rbac-snapshot-controller.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-3.0/deploy/kubernetes/snapshot-
controller/setup-snapshot-controller.yaml
```



CSI Snapshotter fournit un "[validation du crochet](#)" pour aider les utilisateurs à valider les snapshots v1beta1 existants et à confirmer qu'ils sont des objets de ressource valides. Le crochet Web de validation étiquette automatiquement les objets de snapshot non valides et empêche la création d'objets futurs non valides. Le hook validant est déployé par Kubernetes orchestrator. Voir les instructions de déploiement manuel du crochet de validation "[ici](#)". Rechercher des exemples de manifestes d'instantanés non valides "[ici](#)".

L'exemple détaillé ci-dessous décrit les constructions requises pour travailler avec des snapshots et montre comment créer et utiliser des snapshots.

Étape 1 : configurer un `VolumeSnapshotClass`

Avant de créer un snapshot de volume, configurez un lien `../trident-Reference/objects.html[VolumeSnapshotClass^]`.


```
$ cat snap-sc.yaml
#Use apiVersion v1 for Kubernetes 1.20 and above. For Kubernetes 1.17 -
1.19, use apiVersion v1beta1.
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshotClass
metadata:
  name: csi-snapclass
driver: csi.trident.netapp.io
deletionPolicy: Delete
```

Le driver Indique le conducteur CSI d'Astra Trident. `deletionPolicy` peut être `Delete` ou `Retain`. Lorsqu'il est réglé sur `Retain`, le snapshot physique sous-jacent sur le cluster de stockage est conservé même lorsque `VolumeSnapshot` l'objet a été supprimé.

Étape 2 : création d'un snapshot d'un volume persistant existant

```
$ cat snap.yaml
#Use apiVersion v1 for Kubernetes 1.20 and above. For Kubernetes 1.17 -
1.19, use apiVersion v1beta1.
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshot
metadata:
  name: pvcl-snap
spec:
  volumeSnapshotClassName: csi-snapclass
  source:
    persistentVolumeClaimName: pvcl
```

La copie Snapshot est en cours de création pour une demande de volume persistant nommée `pvcl`, et le nom du snapshot est défini sur `pvcl-snap`.

```
$ kubectl create -f snap.yaml
volumesnapshot.snapshot.storage.k8s.io/pvcl-snap created

$ kubectl get volumesnapshots
NAME                AGE
pvcl-snap           50s
```

Cela a créé un `VolumeSnapshot` objet. Un instantané `VolumeSnapshot` est similaire à une demande de volume persistant et est associé à une `VolumeSnapshotContent` objet qui représente le snapshot réel.

Il est possible d'identifier le `VolumeSnapshotContent` objet pour le `pvcl-snap` `VolumeSnapshot` en le décrivant.

```

$ kubectl describe volumesnapshots pvcl-snap
Name:          pvcl-snap
Namespace:    default
.
.
.
Spec:
  Snapshot Class Name:  pvcl-snap
  Snapshot Content Name: snapcontent-e8d8a0ca-9826-11e9-9807-525400f3f660
  Source:
    API Group:
    Kind:      PersistentVolumeClaim
    Name:      pvcl
Status:
  Creation Time:  2019-06-26T15:27:29Z
  Ready To Use:  true
  Restore Size:  3Gi
.
.

```

Le `Snapshot Content Name` identifie l'objet `VolumeSnapshotContent` qui sert ce snapshot. Le `Ready To Use` paramètre indique que l'instantané peut être utilisé pour créer une nouvelle demande de volume persistant.

Étape 3 : création de demandes de volume persistant à partir de copies Snapshot VolumeCas

Pour cela, reportez-vous à l'exemple suivant de création d'une demande de volume persistant à l'aide d'un snapshot :

```

$ 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: pvcl-snap
    kind: VolumeSnapshot
    apiGroup: snapshot.storage.k8s.io

```

`dataSource` La montre que la demande de volume persistant doit être créée à l'aide d'un Snapshot VolumeSnapshot nommé `pvc1-snap` comme source des données. Cela demande à Astra Trident de créer un volume persistant à partir du snapshot. Une fois la demande de volume persistant créée, elle peut être connectée à un pod et utilisée comme n'importe quel autre PVC.



Lors de la suppression d'un volume persistant avec les snapshots associés, le volume Trident correspondant est mis à jour et passe à un état « Suppression ». Pour supprimer le volume Astra Trident, il est nécessaire de supprimer les snapshots du volume.

Trouvez plus d'informations

- "[Snapshots de volume](#)"
- lien `../trident-reference/objects.html[VolumeSnapshotClass^]`

Développement des volumes

Astra Trident permet aux utilisateurs de Kubernetes d'étendre leurs volumes après leur création. Trouvez des informations sur les configurations requises pour développer les volumes iSCSI et NFS.

Développez un volume iSCSI

Vous pouvez développer un volume persistant iSCSI à l'aide du mécanisme de provisionnement CSI.



L'extension de volume iSCSI est prise en charge par `ontap-san`, `ontap-san-economy`, `solidfire-san` Pilotes et requiert Kubernetes 1.16 et version ultérieure.

Présentation

L'extension d'un volume persistant iSCSI comprend les étapes suivantes :

- Modification de la définition de classe de stockage pour définir le `allowVolumeExpansion` champ à `true`.
- Modification de la définition de PVC et mise à jour de `spec.resources.requests.storage` pour refléter la nouvelle taille souhaitée, qui doit être supérieure à la taille d'origine.
- Pour redimensionner le volume persistant, vous devez le connecter à un pod. Lors du redimensionnement d'un volume persistant iSCSI, deux scénarios sont possibles :
 - Si le volume persistant est connecté à un pod, Astra Trident étend le volume en back-end, reanalyse le système et redimensionne le système de fichiers.
 - Pour redimensionner un volume persistant non connecté, Astra Trident étend le volume sur le back-end. Une fois le volume de volume persistant lié à un pod, Trident analyse de nouveau le périphérique et redimensionne le système de fichiers. Kubernetes met ensuite à jour la taille de la demande de volume persistant une fois l'opération d'extension terminée.

L'exemple ci-dessous montre le fonctionnement de l'extension de PV iSCSI.

Étape 1 : configurer la classe de stockage pour prendre en charge l'extension de volume

```

$ 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 une classe de stockage déjà existante, modifiez-la pour l'inclure `allowVolumeExpansion` paramètre.

Étape 2 : créez une demande de volume persistant avec la classe de stockage que vous avez créée

```

$ cat pvc-ontapsan.yaml
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: san-pvc
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: ontap-san

```

Astra Trident crée un volume persistant qui l'associe à cette demande de volume persistant.

```

$ 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 : définissez un pod qui fixe la demande de volume persistant

Dans cet exemple, un pod est créé et utilise le `san-pvc`.

```
$ kubectl get pod
NAME          READY   STATUS    RESTARTS   AGE
centos-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:    centos-pod
```

Étape 4 : développez le volume persistant

Pour redimensionner la PV créée de 1Gi à 2Gi, modifiez la définition de la demande de volume persistant et mettez à jour la `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'extension

Vous pouvez valider le bon fonctionnement de l'extension en contrôlant la taille de la demande de volume persistant, du volume persistant et du volume Astra 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    |
+-----+-----+-----+-----+
+-----+-----+-----+-----+

```

Développez un volume NFS

Astra Trident prend en charge l'extension de volume pour les volumes persistants NFS provisionnés sur ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, aws-cvs, gcp-cvs, et azure-netapp-files systèmes back-end.

Étape 1 : configurer la classe de stockage pour prendre en charge l'extension de volume

Pour redimensionner un volume persistant NFS, l'administrateur doit d'abord configurer la classe de stockage afin de permettre l'extension du volume en paramétrant le `allowVolumeExpansion` champ à `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 permettre l'extension de volume.

Étape 2 : créez une demande de volume persistant avec la classe de stockage que vous avez créée

```
$ cat pvc-ontapnas.yaml
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: ontapnas20mb
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 20Mi
  storageClassName: ontapnas
```

Astra Trident doit créer un volume persistant NFS 20MiB pour cette demande de volume persistant :

```
$ 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 : développez le volume persistant

Pour redimensionner le volume persistant 20MiB nouvellement créé à 1 Gio, modifiez la demande de volume persistant et définissez-la `spec.resources.requests.storage` à 1Go :


```
$ 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'extension

Vous pouvez valider le redimensionnement correctement en contrôlant la taille de la demande de volume persistant, de la volume persistant et du volume Astra 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     |
+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+

```

Importer des volumes

Vous pouvez importer des volumes de stockage existants sous forme de volume persistant Kubernetes à l'aide de `tridentctl import`.

Pilotes prenant en charge l'importation de volumes

Ce tableau décrit les pilotes qui prennent en charge l'importation de volumes et la version dans laquelle ils ont été introduits.

Conducteur	Relâchez
ontap-nas	19.04
ontap-nas-flexgroup	19.04
solidfire-san	19.04
aws-cvs	19.04

Conducteur	Relâchez
azure-netapp-files	19.04
gcp-cvs	19.04
ontap-san	19.04

Pourquoi importer des volumes ?

L'importation d'un volume dans Trident est possible dans plusieurs champs d'application :

- Conaerisation d'une application et réutilisation de son jeu de données existant
- L'utilisation d'un clone du jeu de données pour une application éphémère
- Reconstruction d'un cluster Kubernetes défaillant
- Migration des données applicatives pendant la reprise sur incident

Comment fonctionne l'importation ?

Le fichier de demande de volume persistant est utilisé par le processus d'importation des volumes pour créer la demande de volume persistant. Au minimum, le fichier PVC doit inclure les champs Nom, espace de noms, Access modes et storageClassName, comme indiqué dans l'exemple suivant.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: my_claim
  namespace: my_namespace
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: my_storage_class
```

Le `tridentctl` le client est utilisé pour importer un volume de stockage existant. Trident importe le volume en persistant des métadonnées de volume et en créant la demande de volume persistant et le volume persistant.

```
$ tridentctl import volume <backendName> <volumeName> -f <path-to-pvc-
file>
```

Pour importer un volume de stockage, spécifiez le nom du back-end Astra Trident contenant le volume, ainsi que le nom qui identifie de manière unique le volume sur le stockage (par exemple : ONTAP FlexVol, Element Volume, chemin du volume CVS). Le volume de stockage doit autoriser l'accès en lecture/écriture et être accessible par le back-end Trident spécifié de l'Astra. Le `-f` L'argument de chaîne est requis et spécifie le chemin d'accès au fichier PVC YAML ou JSON.

Lorsque Astra Trident reçoit la demande de volume d'importation, la taille du volume existant est déterminée et définie dans le volume persistant. Une fois le volume importé par le pilote de stockage, le volume persistant est créé avec un SécurRef dans la demande de volume persistant. La règle de récupération est initialement définie sur `retain` Dans la PV. Une fois que Kubernetes a réussi à relier la demande de volume persistant et le volume persistant, la règle de récupération est mise à jour pour correspondre à la règle de récupération de la classe de stockage. Si la règle de récupération de la classe de stockage est `delete`, Le volume de stockage sera supprimé lorsque le volume persistant est supprimé.

Lorsqu'un volume est importé avec le `--no-manage` Argument, Trident n'effectue aucune opération supplémentaire sur la demande de volume persistant ou la volume persistant pour le cycle de vie des objets. Trident ignore les événements PV et PVC pour `--no-manage` Objets, le volume de stockage n'est pas supprimé lors de la suppression du volume persistant. D'autres opérations, telles que le clone de volume et le redimensionnement des volumes, sont également ignorées. Cette option est utile si vous souhaitez utiliser Kubernetes pour des workloads conteneurisés, mais que vous souhaitez gérer le cycle de vie du volume de stockage en dehors de Kubernetes.

Une annotation est ajoutée pour la demande de volume persistant et la volume persistant, qui servent un double objectif : indiquer l'importation du volume et gérer la demande de volume persistant. Cette annotation ne doit pas être modifiée ni supprimée.

Trident 19.07 et les versions ultérieures traitent la pièce jointe des volumes persistants et monte le volume dans le cadre de son importation. Pour les importations utilisant les versions antérieures d'Astra Trident, il n'y aura aucune opération dans le chemin d'accès aux données et l'importation de volume ne vérifiera pas si le volume peut être monté. En cas d'erreur lors de l'importation de volumes (par exemple, la classe de stockage n'est pas correcte), vous pouvez effectuer une restauration en changeant la règle de récupération du volume persistant à `retain`, Suppression de la demande de volume persistant et nouvelle tentative de la commande d'importation du volume.

`ontap-nas` **et** `ontap-nas-flexgroup` **importations**

Chaque volume créé avec le `ontap-nas` Le pilote est un FlexVol sur le cluster ONTAP. Importation de volumes FlexVol avec `ontap-nas` le pilote fonctionne de la même manière. Une FlexVol qui existe déjà sur un cluster ONTAP peut être importée en tant que `ontap-nas PVC`. De même, les volumes FlexGroup peuvent être importés en tant que `ontap-nas-flexgroup ESV`.



Pour être importé par Trident, un volume ONTAP doit être de type `rw`. Si un volume est de type `dp`, il s'agit d'un volume de destination SnapMirror ; vous devez rompre la relation du miroir avant d'importer le volume dans Trident.



Le `ontap-nas` le pilote ne peut pas importer et gérer les qtrees. Le `ontap-nas` et `ontap-nas-flexgroup` les pilotes n'autorisent pas les noms de volumes dupliqués.

Par exemple, pour importer un volume nommé `managed_volume` sur un système back-end nommé `ontap_nas`, utilisez la commande suivante :

```
$ tridentctl import volume ontap_nas managed_volume -f <path-to-pvc-file>
```

PROTOCOL	NAME	BACKEND UUID	SIZE	STORAGE CLASS	STATE	MANAGED
file	pvc-bf5ad463-afbb-11e9-8d9f-5254004dfdb7	c5a6f6a4-b052-423b-80d4-8fb491a14a22	1.0 GiB	standard	online	true

Pour importer un volume nommé `unmanaged_volume` (sur le `ontap_nas` backend), que Trident ne gère pas, utilisez la commande suivante :

```
$ tridentctl import volume nas_blog unmanaged_volume -f <path-to-pvc-file>
--no-manage
```

PROTOCOL	NAME	BACKEND UUID	SIZE	STORAGE CLASS	STATE	MANAGED
file	pvc-df07d542-afbc-11e9-8d9f-5254004dfdb7	c5a6f6a4-b052-423b-80d4-8fb491a14a22	1.0 GiB	standard	online	false

Lorsque vous utilisez le `--no-manage` Argument, Trident ne renomme pas le volume ni ne valide si le volume a été monté. L'opération d'importation du volume échoue si le volume n'a pas été monté manuellement.



Un bogue existant précédemment avec l'importation de volumes avec Unixpermissions personnalisées a été corrigé. Vous pouvez spécifier `unixpermissions` dans votre définition de PVC ou configuration back-end et demander à Astra Trident d'importer le volume en conséquence.

ontap-san importer

Astra Trident peut également importer des volumes FlexVol SAN de ONTAP contenant un seul LUN. Ceci est cohérent avec le `ontap-san` Pilote, qui crée un FlexVol pour chaque demande de volume persistant et une LUN au sein de la FlexVol. Vous pouvez utiliser le `tridentctl import` commande de la même manière que dans les autres cas :

- Inclure le nom du `ontap-san` back-end.


```
$ 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   |
+-----+-----+-----+
+-----+-----+-----+-----+
```



Le pilote d'élément prend en charge les noms de volume dupliqués. S'il existe des noms de volume dupliqués, le processus d'importation de volume de Trident renvoie une erreur. Pour contourner ce problème, clonez le volume et fournissez un nom de volume unique. Importez ensuite le volume cloné.

aws-cvs importer



Pour importer un volume sauvegardé par NetApp Cloud Volumes Service dans AWS, identifiez le volume par son chemin d'accès au volume et non son nom.

Pour importer un `aws-cvs` volume sur le back-end appelé `awscvs_YEppr` avec le chemin de volume de `adroit-jolly-swift`, utilisez la commande suivante :

```
$ tridentctl import volume awscvs_YEppr adroit-jolly-swift -f <path-to-pvc-file> -n trident
```

```
+-----+-----+-----+
+-----+-----+-----+-----+
|          NAME          | SIZE | STORAGE CLASS |
PROTOCOL |          BACKEND UUID          | STATE | MANAGED |
+-----+-----+-----+
+-----+-----+-----+-----+
| pvc-a46ccab7-44aa-4433-94b1-e47fc8c0fa55 | 93 GiB | aws-storage   | file
| e1a6e65b-299e-4568-ad05-4f0a105c888f | online | true         |
+-----+-----+-----+
+-----+-----+-----+-----+
```



Le chemin du volume correspond à la partie du chemin d'exportation du volume après `./`. Par exemple, si le chemin d'exportation est `10.0.0.1:/adroit-jolly-swift`, le chemin du volume est `adroit-jolly-swift`.

gcp-cvs importer

Importation d'un gcp-cvs le volume fonctionne de la même manière que l'importation d'un aws-cvs volumétrie.

azure-netapp-files importer

Pour importer un azure-netapp-files volume sur le back-end appelé azurenetaappfiles_40517 avec le chemin de volume importvoll1, exécutez la commande suivante :

```
$ tridentctl import volume azurenetaappfiles_40517 importvoll1 -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      |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
```



Le chemin de volume du volume ANF est présent dans le chemin de montage après :/. Par exemple, si le chemin de montage est 10.0.0.2:/importvoll1, le chemin du volume est importvoll1.

Préparez le nœud de travail

Tous les nœuds workers du cluster Kubernetes doivent pouvoir monter les volumes provisionnés pour vos pods. Si vous utilisez le ontap-nas, ontap-nas-economy, ou ontap-nas-flexgroup Pilote pour l'un de vos systèmes back-end, les nœuds workers ont besoin des outils NFS. Sinon, ils nécessitent les outils iSCSI.

Les versions récentes de Red Hat CoreOS disposent par défaut de NFS et d'iSCSI.



Redémarrez toujours les nœuds workers après l'installation des outils NFS ou iSCSI, sinon la connexion de volumes à ces conteneurs risque de tomber en panne.

Volumes NFS

Protocole	Système d'exploitation	Commandes
NFS	RHEL/CentOS	sudo yum install -y nfs-utils

Protocole	Système d'exploitation	Commandes
NFS	Ubuntu/Debian	sudo apt-get install -y nfs-common



Assurez-vous que le service NFS est démarré pendant le démarrage.


Volumes iSCSI

Tenez compte des points suivants lorsque vous utilisez des volumes iSCSI :

- Chaque nœud du cluster Kubernetes doit avoir un IQN unique. **C'est une condition préalable nécessaire.**
- Si vous utilisez RHCOS version 4.5 ou ultérieure, ou RHEL ou CentOS version 8.2 ou ultérieure avec `solidfire-san Pilote`, assurez-vous que l'algorithme d'authentification CHAP est défini sur MD5 dans `/etc/iscsi/iscsid.conf`.

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

- Lorsque vous utilisez des nœuds workers exécutant RHEL/RedHat CoreOS avec iSCSI PVS, veillez à spécifier le `discard MounOption` dans la classe de stockage pour effectuer la réclamation d'espace en ligne. Voir "[La documentation de Red Hat](#)".

Protocole	Système d'exploitation	Commandes
ISCSI	RHEL/CentOS	<p>1. Installez les packages système suivants :</p> <pre>sudo yum install -y lsscsi iscsi-initiator- utils sg3_utils device- mapper-multipath</pre> <p>2. Vérifiez que la version iscsi-initiator-utils est 6.2.0.874-2.el7 ou ultérieure :</p> <pre>rpm -q iscsi-initiator- utils</pre> <p>3. Définir la numérisation sur manuelle :</p> <pre>sudo sed -i 's/^\(node.session.scan \).*\/\1 = manual/' /etc/iscsi/iscsid.conf</pre> <p>4. Activer les chemins d'accès multiples :</p> <pre>sudo mpathconf --enable --with_multipathd y --find_multipaths n</pre> <div style="border: 1px solid #ccc; padding: 5px; margin: 10px 0;"> <p> Bien sûr etc/multipath.conf contient find_multipaths no sous defaults.</p> </div> <p>5. S'assurer que iscsid et multipathd sont en cours d'exécution :</p> <pre>sudo systemctl enable --now iscsid multipathd</pre> <p>6. Activer et démarrer iscsi:</p> <pre>sudo systemctl enable --now iscsi</pre>

Protocole	Système d'exploitation	Commandes
ISCSI	Ubuntu/Debian	<p>1. Installez les packages système suivants :</p> <pre>sudo apt-get install -y open-iscsi lsscsi sg3-utils multipath-tools scsitools</pre> <p>2. Vérifiez que la version Open-iscsi est 2.0.874-5ubuntu2.10 ou ultérieure (pour bionique) ou 2.0.874-7.1ubuntu6.1 ou ultérieure (pour focaux) :</p> <pre>dpkg -l open-iscsi</pre> <p>3. Définir la numérisation sur manuelle :</p> <pre>sudo sed -i 's/^\(node.session.scan\).*\/\1 = manual/' /etc/iscsi/iscsid.conf</pre> <p>4. Activer les chemins d'accès multiples :</p> <pre>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</pre> <div style="border: 1px solid gray; padding: 5px; margin: 10px 0;"> <p> Bien sûr etc/multipath.conf contient find_multipaths no sous defaults.</p> </div> <p>5. S'assurer que open-iscsi et multipath-tools sont activées et en cours d'exécution :</p> <pre>sudo systemctl status 121 multipath-tools</pre>



Pour Ubuntu 18.04, vous devez découvrir les ports cibles avec `iscsiadm` avant de commencer `open-iscsi`. Pour que le démon iSCSI démarre, vous pouvez également modifier le `iscsi` service à démarrer `iscsid` automatiquement.

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



Pour en savoir plus sur la préparation automatique des nœuds workers, qui est une fonctionnalité bêta, reportez-vous à la section "[ici](#)".

Préparation automatique du nœud de travail

Astra Trident peut automatiquement installer le requis NFS et iSCSI Outils sur les nœuds présents dans le cluster Kubernetes. Il s'agit d'une fonction **bêta** et est **non destiné aux grappes de production**. Aujourd'hui, cette fonction est disponible pour les nœuds qui exécutent **CentOS, RHEL et Ubuntu**.

Pour ce faire, Astra Trident inclut un nouveau drapeau d'installation : `--enable-node-prep` pour les installations déployées avec `tridentctl`. Pour les déploiements avec l'opérateur Trident, utilisez l'option booléenne `enableNodePrep`.



Le `--enable-node-prep` L'option d'installation permet à Astra Trident d'installer et de s'assurer que les packages et/ou services NFS et iSCSI sont exécutés lorsqu'un volume est monté sur un nœud worker. Il s'agit d'une fonction **bêta** destinée à être utilisée dans les environnements de développement/test qui est **non qualifié** pour la production.

Lorsque le `--enable-node-prep` Flag est inclus dans les installations Trident d'Astra déployées avec `tridentctl`, voici ce qui se passe:

1. Lors de l'installation, Astra Trident enregistre les nœuds sur lesquels il s'exécute.
2. Lorsqu'une demande de demande de demande de volume persistant est formulée, Astra Trident crée un volume persistant à partir de l'un des systèmes back-end gérés.
3. Pour utiliser la demande de volume persistant dans un pod, Astra Trident doit monter le volume sur le nœud sur lequel le pod s'exécute. Astra Trident tente d'installer les utilitaires clients NFS/iSCSI requis et de s'assurer que les services requis sont actifs. Cette opération s'effectue avant le montage du volume.

La préparation d'un nœud de travail s'effectue une seule fois dans le cadre de la première tentative de montage d'un volume. Tous les montages de volume suivants doivent réussir tant que les modifications ne sont pas apportées en dehors d'Astra Trident NFS et iSCSI utilitaires.

Ainsi, Astra Trident peut s'assurer que tous les nœuds d'un cluster Kubernetes disposent des utilitaires nécessaires au montage et à la connexion des volumes. Pour les volumes NFS, l'export policy doit également permettre le montage du volume. Trident peut gérer automatiquement les règles d'exportation par système back-end. Les utilisateurs peuvent également gérer les règles d'exportation hors bande.

Contrôle d'Astra Trident

Astra Trident fournit un ensemble de terminaux de metrics de Prometheus que vous pouvez utiliser pour surveiller les performances d'Astra Trident.

Avec les metrics d'Astra Trident, vous pouvez :

- Surveillez l'état et la configuration d'Astra Trident. Vous avez la possibilité d'examiner la réussite des opérations et de savoir si elles peuvent communiquer avec les systèmes back-end comme prévu.

- Examiner les informations d'utilisation du système back-end et comprendre le nombre de volumes provisionnés sur un système back-end, ainsi que la quantité d'espace consommé, etc.
- Conservez un mappage de la quantité de volumes provisionnés sur les systèmes back-end disponibles.
- Suivi des performances. Découvrez le temps nécessaire pour communiquer avec Astra Trident aux systèmes back-end et effectuer les opérations.



Par défaut, les metrics de Trident sont visibles sur le port cible 8001 au `/metrics` point final. Ces mesures sont **activées par défaut** lors de l'installation de Trident.

Ce dont vous avez besoin

- Cluster Kubernetes avec Astra Trident installé.
- Instance Prometheus. Il peut s'agir d'un ["Déploiement conteneurisé par Prometheus"](#) Vous pouvez également utiliser Prometheus en tant que ["application native"](#).

Étape 1 : définir une cible Prometheus

Vous devez définir une cible Prometheus pour collecter les metrics et obtenir des informations sur les systèmes back-end gérés par Trident, les volumes qu'elle crée, etc. C'est ça ["Blog"](#) Vous apprendrez comment utiliser Prometheus et Grafana avec Astra Trident pour récupérer des metrics. Ce blog explique comment exécuter Prometheus en tant qu'opérateur dans votre cluster Kubernetes et créer un ServiceMonitor pour obtenir les metrics d'Astra Trident.

Étape 2 : créer un ServiceMonitor Prometheus

Pour consommer les metrics Trident, vous devez créer un ServiceMonitor Prometheus qui surveille la `trident-csi` service et écoute sur le `metrics` port. Un exemple de ServiceMonitor se présente comme suit :

```
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  name: trident-sm
  namespace: monitoring
  labels:
    release: prom-operator
spec:
  jobLabel: trident
  selector:
    matchLabels:
      app: controller.csi.trident.netapp.io
  namespaceSelector:
    matchNames:
      - trident
  endpoints:
    - port: metrics
      interval: 15s
```

Cette définition de ServiceMonitor récupère les mesures renvoyées par le `trident-csi` et recherche spécifiquement le `metrics` point d'extrémité du service. Par conséquent, Prometheus est désormais configuré pour être en mesure de comprendre les `metrics` d'Astra Trident.

Outre les `metrics` directement disponibles par Astra Trident, kubelet expose beaucoup `kubelet_volume_*` `metrics` via son propre terminal de `metrics`. Kubelet peut fournir des informations sur les volumes reliés, ainsi que sur les pods et autres opérations internes qu'elle gère. Voir ["ici"](#).

Étape 3 : interroger les mesures Trident avec PromQL

PromQL est bon pour la création d'expressions qui renvoient des séries chronologiques ou des données tabulaires.

Voici quelques questions PromQL que vous pouvez utiliser :

Accédez aux informations sur l'état de santé de Trident

- **Pourcentage de réponses HTTP 2XX d'Astra Trident**

```
(sum (trident_rest_ops_seconds_total_count{status_code=~"2.."} OR on()  
vector(0)) / sum (trident_rest_ops_seconds_total_count)) * 100
```

- **Pourcentage de réponses REST d'Astra Trident par le code d'état**

```
(sum (trident_rest_ops_seconds_total_count) by (status_code) / scalar  
(sum (trident_rest_ops_seconds_total_count))) * 100
```

- **Durée moyenne en ms des opérations effectuées par Astra Trident**

```
sum by (operation)  
(trident_operation_duration_milliseconds_sum{success="true"}) / sum by  
(operation)  
(trident_operation_duration_milliseconds_count{success="true"})
```

Découvrez les informations d'utilisation d'Astra Trident

- **Taille moyenne du volume**

```
trident_volume_allocated_bytes/trident_volume_count
```

- **Espace volume total provisionné par chaque back-end**

```
sum (trident_volume_allocated_bytes) by (backend_uuid)
```

Utiliser individuellement le volume



Cette activation est uniquement possible si les indicateurs kubelet sont également collectés.

- **Pourcentage d'espace utilisé pour chaque volume**

```
kubelet_volume_stats_used_bytes / kubelet_volume_stats_capacity_bytes * 100
```

Découvrez la télémétrie AutoSupport d'Astra Trident

Par défaut, Astra Trident envoie des metrics de Prometheus et des informations de base back-end à NetApp dans un rythme quotidien.

- Pour empêcher Astra Trident d'envoyer des metrics de Prometheus et des informations de base back-end à NetApp, passez le `--silence-autosupport` Indicateur lors de l'installation d'Astra Trident.
- Astra Trident peut également envoyer des journaux de conteneur à NetApp support à la demande via `tridentctl send autosupport`. Vous devrez déclencher Astra Trident pour télécharger ses journaux. Avant de communiquer les journaux, vous devez accepter celui de NetApp <https://www.netapp.com/company/legal/privacy-policy/>["politique de confidentialité"].
- Sauf mention contraire, Astra Trident extrait les journaux des 24 dernières heures.
- Vous pouvez spécifier la période de conservation des journaux à l'aide de l' `--since` drapeau. Par exemple : `tridentctl send autosupport --since=1h`. Ces informations sont collectées et envoyées via un `trident-autosupport` Conteneur installé avec Astra Trident. Vous pouvez obtenir l'image conteneur à "[AutoSupport Trident](#)".
- Le AutoSupport Trident ne collecte pas et ne transmet pas d'informations à caractère personnel (PII) ou de données personnelles. Il est livré avec un "[CLUF](#)" Ce n'est pas applicable à l'image du conteneur Trident elle-même. Pour en savoir plus sur l'engagement de NetApp en matière de sécurité des données et de confiance "[ici](#)".

Voici un exemple de charge utile envoyée par Astra Trident :

```

{
  "items": [
    {
      "backendUUID": "ff3852e1-18a5-4df4-b2d3-f59f829627ed",
      "protocol": "file",
      "config": {
        "version": 1,
        "storageDriverName": "ontap-nas",
        "debug": false,
        "debugTraceFlags": null,
        "disableDelete": false,
        "serialNumbers": [
          "nwkvzfanek_SN"
        ],
        "limitVolumeSize": ""
      },
      "state": "online",
      "online": true
    }
  ]
}

```

- Les messages AutoSupport sont envoyés au terminal AutoSupport de NetApp. Si vous utilisez un registre privé pour stocker des images de conteneur, vous pouvez utiliser le `--image-registry` drapeau.
- Vous pouvez également configurer des URL proxy en générant les fichiers YAML d'installation. Pour ce faire, utilisez `tridentctl install --generate-custom-yaml` Pour créer les fichiers YAML et ajouter le `--proxy-url` argument pour le `trident-autosupport` conteneur `trident-deployment.yaml`.

Désactivation des metrics d'Astra Trident

Pour désactiver** les mesures signalées, vous devez générer des YAML personnalisées (à l'aide de `--generate-custom-yaml` marquer) et modifiez-les pour supprimer le `--metrics` indicateur d'être appelé pour le `trident-main` conteneur.

Informations sur le copyright

Copyright © 2024 NetApp, Inc. Tous droits réservés. Imprimé aux États-Unis. Aucune partie de ce document protégé par copyright ne peut être reproduite sous quelque forme que ce soit ou selon quelque méthode que ce soit (graphique, électronique ou mécanique, notamment par photocopie, enregistrement ou stockage dans un système de récupération électronique) sans l'autorisation écrite préalable du détenteur du droit de copyright.

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

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

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

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

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

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

Informations sur les marques commerciales

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