Référence

Trident

NetApp
January 14, 2026

Sommaire

Référence

Ports Trident

En savoir plus sur les ports utilisés par Trident pour la communication.

Ports Trident

Trident communique sur les ports suivants :

Port	Objectif
8443	HTTPS backChannel
8001	Terminal des metrics Prometheus
8000	Serveur REST Trident
17546	Port de sonde de liaison/préparation utilisé par les modules de démonset Trident

Le port de la sonde d'état de disponibilité/préparation peut être modifié lors de l'installation à l'aide de `--probe-port` l'indicateur. Il est important de s'assurer que ce port n'est pas utilisé par un autre processus sur les nœuds worker.

API REST Trident

Sont le moyen le plus simple d'interagir avec l'API REST Trident, mais "commandes et options tridentctl"vous pouvez utiliser le terminal REST directement si vous préférez.

Quand utiliser l'API REST

L'API REST est utile pour les installations avancées qui utilisent Trident en tant que fichier binaire autonome dans les déploiements non Kubernetes.

Pour une meilleure sécurité, Trident `REST API` est limité par défaut à localhost lors de l'exécution dans un pod. Pour modifier ce comportement, vous devez définir l'argument de Trident `-address` dans sa configuration de pod.

Avec l'API REST

Pour des exemples de la façon dont ces API sont appelées, passez (`-d`l'indicateur debug). Pour plus d'informations, reportez-vous "Gérez Trident à l'aide de tridentctl"à .

L'API fonctionne comme suit :

OBTENEZ

`GET <trident-address>/trident/v1/<object-type>`

Répertorie tous les objets de ce type.

GET <trident-address>/trident/v1/<object-type>/<object-name>

Obtient les détails de l'objet nommé.

POST

POST <trident-address>/trident/v1/<object-type>

Crée un objet du type spécifié.

- Nécessite une configuration JSON pour que l'objet soit créé. Pour la spécification de chaque type d'objet, reportez-vous "Gérez Trident à l'aide de tridentctl"à la .
- Si l'objet existe déjà, le comportement varie : les systèmes back-end mettent à jour l'objet existant, tandis que tous les autres types d'objet échoueront.

SUPPRIMER

DELETE <trident-address>/trident/v1/<object-type>/<object-name>

Supprime la ressource nommée.

> Les volumes associés aux systèmes back-end ou aux classes de stockage continueront d'exister. Ils doivent être supprimés séparément. Pour plus d'informations, reportez-vous "Gérez Trident à l'aide de tridentctl"à .

Options de ligne de commande

Trident expose plusieurs options de ligne de commande pour l'orchestrateur Trident. Vous pouvez utiliser ces options pour modifier votre déploiement.

Journalisation

-debug

Active la sortie de débogage.

-loglevel <level>

Définit le niveau de journalisation (débogage, info, avertissement, erreur, fatal). La valeur par défaut est INFO.

Kubernetes

-k8s_pod

Utilisez cette option ou `-k8s_api_server` pour activer la prise en charge de Kubernetes. La configuration de cette configuration entraîne l'utilisation par Trident des identifiants du compte de service Kubernetes du pod qui y est associé pour contacter le serveur d'API. Cela fonctionne uniquement lorsque Trident s'exécute en tant que pod dans un cluster Kubernetes avec les comptes de service activés.

-k8s_api_server <insecure-address:insecure-port>

Utilisez cette option ou `-k8s_pod` pour activer la prise en charge de Kubernetes. Lorsqu'il est spécifié, Trident se connecte au serveur API Kubernetes à l'aide de l'adresse et du port non sécurisés fournis. Cela permet de déployer Trident en dehors d'un pod. Cependant, il ne prend en charge que les connexions non sécurisées au serveur d'API. Pour vous connecter en toute sécurité, déployez Trident dans un pod avec l' `-

k8s_pod`option.

Docker

-volume_driver <name>

Nom du pilote utilisé lors de l'enregistrement du plug-in Docker. La valeur par défaut est `netapp` .

-driver_port <port-number>

Écoutez sur ce port plutôt que sur un socket de domaine UNIX.

-config <file>

Obligatoire ; vous devez spécifier ce chemin vers un fichier de configuration back-end.

REPOS

-address <ip-or-host>

Spécifie l'adresse à laquelle le serveur REST de Trident doit écouter. Par défaut, localhost. Lorsque vous écoutez sur localhost et exécutez-les dans un pod Kubernetes, l'interface REST n'est pas directement accessible depuis l'extérieur du pod. `-address ""`Permet de rendre l'interface REST accessible à partir de l'adresse IP du pod.

Vous pouvez configurer l'interface REST de Trident pour écouter et utiliser l'interface 127.0.0.1 (pour IPv4) ou [::1] (pour IPv6) uniquement.

-port <port-number>

Spécifie le port sur lequel le serveur REST de Trident doit écouter. La valeur par défaut est 8000.

-rest

Active l'interface REST. Valeur true par défaut.

Kubernetes et objets Trident

Vous pouvez interagir avec Kubernetes et Trident à l'aide des API REST en lisant et en écrivant des objets de ressource. La relation entre Kubernetes et Trident, Trident et le stockage, ainsi que Kubernetes et le stockage est établie avec plusieurs objets de ressources. Certains de ces objets sont gérés par Kubernetes et d'autres sont gérés à l'aide de Trident.

Comment les objets interagissent-ils les uns avec les autres ?

La manière la plus simple de comprendre les objets, leur rôle et leur interaction consiste à suivre une seule demande de stockage auprès d'un utilisateur Kubernetes :

1. Un utilisateur crée une `PersistentVolumeClaim` demande de nouvelle d' `PersistentVolume`une taille particulière à partir d'un Kubernetes `StorageClass` précédemment configuré par l'administrateur.
2. Kubernetes `StorageClass` identifie Trident comme provisionneur et inclut des paramètres qui indiquent à Trident comment provisionner un volume pour la classe demandée.

3. Trident se voit attribuer le `StorageClass` même nom qui identifie la correspondance `Backends` et `StoragePools` qu'il peut utiliser pour provisionner des volumes pour la classe.
4. Trident provisionne le stockage sur un back-end correspondant et crée deux objets : un `PersistentVolume` dans Kubernetes qui indique à Kubernetes comment rechercher, monter et traiter le volume, et un volume dans Trident qui conserve la relation entre le `PersistentVolume` et le stockage réel.
5. Kubernetes lie le `PersistentVolumeClaim` au nouveau `PersistentVolume`. Les pods incluant le `PersistentVolumeClaim` montage de ce volume persistant sur tout hôte sur lequel il s'exécute.
6. Un utilisateur crée un `VolumeSnapshot` d'une demande de volume persistant existante, en utilisant un `VolumeSnapshotClass` qui pointe vers Trident.
7. Trident identifie le volume associé à la demande de volume persistant et crée un snapshot du volume sur son back-end. Elle crée également un `VolumeSnapshotContent` qui indique à Kubernetes comment identifier le Snapshot.
8. Un utilisateur peut créer un `PersistentVolumeClaim` utilisant `VolumeSnapshot` comme source.
9. Trident identifie le snapshot requis et exécute le même ensemble d'étapes que celles impliquées dans la création d'un `PersistentVolume` et d'un `Volume`.

Pour en savoir plus sur les objets Kubernetes, nous vous recommandons vivement de lire la "Volumes persistants" section de la documentation Kubernetes.

Objets Kubernetes `PersistentVolumeClaim`

Un objet Kubernetes `PersistentVolumeClaim` est une demande de stockage émise par un utilisateur de cluster Kubernetes.

Outre la spécification standard, Trident permet aux utilisateurs de spécifier les annotations spécifiques au volume suivantes s'ils veulent remplacer les valeurs par défaut que vous définissez dans la configuration back-end :

Annotation	Option de volume	Pilotes pris en charge
trident.netapp.io/fileSystem	Système de fichiers	ontap-san, solidfire-san, ontap-san-economy
trident.netapp.io/cloneFromPVC	Volume cloneSourceVolume	ontap-nas, ontap-san, solidfire-san, azure-netapp-files, gcp-cvs ontap-san-économie
trident.netapp.io/splitOnClone	SplitOnClone	ontap-nas, ontap-san
trident.netapp.io/protocol	protocole	toutes
trident.netapp.io/exportPolicy	ExportPolicy	ontap-nas, économie ontap-nas, ontap-nas-flexgroup
trident.netapp.io/snapshotPolicy	Politique de snapshots	ontap-nas, ontap-nas-économie, ontap-nas-flexgroup, ontap-san
trident.netapp.io/snapshotReserve	Réserve de snapshots	ontap-nas, ontap-nas-flexgroup, ontap-san, gcp-cvs

Annotation	Option de volume	Pilotes pris en charge
trident.netapp.io/snapshotDirectory	Répertoire de snapshots	ontap-nas, économie ontap-nas, ontap-nas-flexgroup
trident.netapp.io/unixPermissions	Autorisations unix	ontap-nas, économie ontap-nas, ontap-nas-flexgroup
trident.netapp.io/blockSize	Taille de bloc	solidfire-san

Si le volume persistant créé est associé à la `Delete` règle de récupération, Trident supprime le volume persistant et le volume de sauvegarde lorsque le volume persistant est libéré (c'est-à-dire lorsque l'utilisateur supprime la demande de volume persistant). En cas d'échec de l'action de suppression, Trident marque le volume persistant comme tel et tente régulièrement l'opération jusqu'à ce qu'il réussisse ou que le volume persistant soit supprimé manuellement. Si le volume persistant utilise `Retain` la règle, Trident l'ignore et suppose que l'administrateur le nettoie depuis Kubernetes et le back-end, ce qui permet de sauvegarder ou d'inspecter le volume avant sa suppression. Notez que la suppression du volume persistant n'entraîne pas la suppression du volume de sauvegarde par Trident. Vous devez le supprimer à l'aide de l'API REST (`tridentctl`).

Trident prend en charge la création de copies Snapshot de volumes à l'aide de la spécification CSI : vous pouvez créer un Snapshot de volume et l'utiliser comme source de données pour cloner des demandes de volume existantes. Ainsi, des copies instantanées de volumes persistants peuvent être exposées à Kubernetes sous forme de snapshots. Les snapshots peuvent ensuite être utilisés pour créer de nouveaux volumes persistants. Jetez un coup d'œil `On-Demand Volume Snapshots` à pour voir comment cela fonctionne.

Trident propose également les `cloneFromPVC` annotations et `splitOnClone` pour la création de clones. Vous pouvez utiliser ces annotations pour cloner une demande de volume persistant sans avoir à utiliser l'implémentation CSI.

Voici un exemple : si un utilisateur a déjà un PVC appelé `mysql`, l'utilisateur peut créer un nouveau PVC appelé à `mysqlclone` l'aide de l'annotation, telle que `trident.netapp.io/cloneFromPVC: mysql`. Avec ce jeu d'annotations, Trident clone le volume correspondant à la demande de volume mysql au lieu de provisionner un volume entièrement.

Prenez en compte les points suivants :

- NetApp recommande de cloner un volume inactif.
- Un volume persistant et son clone doivent se trouver dans le même namespace Kubernetes et avoir la même classe de stockage.
- Avec les `ontap-nas` pilotes et `ontap-san`, il peut être souhaitable de définir l'annotation PVC `trident.netapp.io/splitOnClone` en conjonction avec `trident.netapp.io/cloneFromPVC`. Avec la `trident.netapp.io/splitOnClone` valeur définie sur `true`, Trident divise le volume cloné du volume parent et, par conséquent, dissocie complètement le cycle de vie du volume cloné de son volume parent au détriment de la perte d'efficacité du stockage. L'impossibilité de `trident.netapp.io/splitOnClone` la configurer ou de la configurer sur `false` entraîne une réduction de la consommation d'espace sur le back-end, au détriment de la création de dépendances entre les volumes parent et clone, de sorte que le volume parent ne peut pas être supprimé, sauf si le clone est supprimé en premier. Si le fractionnement du clone s'avère judicieux, il s'agit de cloner un volume de base de données vide où l'on peut attendre du volume et de son clone pour diverger considérablement, et ne bénéficier pas des fonctionnalités d'efficacité du stockage offertes par ONTAP.

Le `sample-input` répertoire contient des exemples de définitions de PVC à utiliser avec Trident. Reportez-

vous à la pour une description complète des paramètres et des paramètres associés aux volumes Trident.

Objets Kubernetes `PersistentVolume`

Un objet Kubernetes `PersistentVolume` représente un élément de stockage mis à disposition du cluster Kubernetes. Il dispose d'un cycle de vie indépendant du pod qui l'utilise.

> Trident crée `PersistentVolume` des objets et les enregistre automatiquement dans le cluster Kubernetes en fonction des volumes qu'il provisionne. Vous n'êtes pas censé les gérer vous-même.

Lorsque vous créez une demande de volume virtuel qui fait référence à un volume basé sur Trident `StorageClass` , Trident provisionne un nouveau volume en utilisant la classe de stockage correspondante et enregistre un nouveau volume persistant pour ce volume. Lors de la configuration du volume provisionné et du volume persistant correspondant, Trident respecte les règles suivantes :

- Trident génère un nom de volume persistant pour Kubernetes et un nom interne utilisé pour le provisionnement du stockage. Dans les deux cas, il garantit que les noms sont uniques dans leur périmètre.
- La taille du volume correspond le plus possible à la taille demandée dans le PVC, bien qu'elle puisse être arrondie à la quantité la plus proche, selon la plate-forme.

Objets Kubernetes `StorageClass`

Les objets Kubernetes `StorageClass` sont spécifiés par leur nom dans `PersistentVolumeClaims` pour provisionner le stockage avec un ensemble de propriétés. La classe de stockage elle-même identifie le mécanisme de provisionnement à utiliser et définit cet ensemble de propriétés, comme le mécanisme de provisionnement le comprend.

Il s'agit de l'un des deux objets de base qui doivent être créés et gérés par l'administrateur. L'autre est l'objet back-end Trident.

Voici à quoi ressemble un objet Kubernetes `StorageClass` utilisant Trident :

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: <Name>
provisioner: csi.trident.netapp.io
mountOptions: <Mount Options>
parameters: <Trident Parameters>
allowVolumeExpansion: true
volumeBindingMode: Immediate
```

Ces paramètres sont spécifiques à Trident et indiquent à Trident comment provisionner des volumes pour la classe.

Les paramètres de classe de stockage sont les suivants :

Attribut	Type	Obligatoire	Description
attributs	chaîne map[string]	non	Voir la section attributs ci-dessous
StoragePools	Mapper[string]StringList	non	Mappage des noms backend avec les listes de pools de stockage dans
Des médutiquesde stockage	Mapper[string]StringList	non	Mappage des noms backend avec les listes de pools de stockage dans
Exclus du stockagePools	Mapper[string]StringList	non	Mappage des noms backend avec les listes de pools de stockage dans

Les attributs de stockage et leurs valeurs possibles peuvent être classés en attributs de sélection des pools de stockage et en attributs Kubernetes.

Attributs de sélection du pool de stockage

Ces paramètres déterminent quels pools de stockage gérés par Trident doivent être utilisés pour provisionner les volumes d'un type donné.

Attribut	Type	Valeurs	Offre	Demande	Pris en charge par
support[1]	chaîne	hdd, hybride, ssd	Le pool contient des supports de ce type ; hybride signifie les deux	Type de support spécifié	ontap-nas, ontap-nas-économie, ontap-nas-flexgroup, ontap-san, solidfire-san
Type de provisionnement	chaîne	fin, épais	Le pool prend en charge cette méthode de provisionnement	Méthode de provisionnement spécifiée	thick : tous les systèmes ONTAP ; thin : tous les systèmes ONTAP et solidfire-san
Type de dos	chaîne	ontap-nas, économie ontap-nas, ontap-nas-flexgroup, ontap-san, solidfire-san, gcp-cvs, azure-netapp-files, ontap-san-economy	Le pool appartient à ce type de système back-end	Backend spécifié	Tous les conducteurs

Attribut	Type	Valeurs	Offre	Demande	Pris en charge par
snapshots	bool	vrai, faux	Le pool prend en charge les volumes dotés de snapshots	Volume sur lequel les snapshots sont activés	ontap-nas, ontap-san, solidfire-san, gcp-cvs
clones	bool	vrai, faux	Le pool prend en charge les volumes de clonage	Volume sur lequel les clones sont activés	ontap-nas, ontap-san, solidfire-san, gcp-cvs
le cryptage	bool	vrai, faux	Le pool prend en charge les volumes chiffrés	Volume avec chiffrement activé	ontap-nas, économie ontap-nas, ontap-nas-flexgroups, ontap-san
LES IOPS	int	entier positif	Le pool est en mesure de garantir l'IOPS dans cette plage	Volume garanti ces IOPS	solidfire-san

[1] : non pris en charge par les systèmes ONTAP Select

Dans la plupart des cas, les valeurs demandées influencent directement le provisionnement ; par exemple, la demande d'un provisionnement lourd entraîne un volume approvisionné. Un pool de stockage Element utilise ses IOPS minimales et maximales pour définir des valeurs de QoS plutôt que la valeur demandée. Dans ce cas, la valeur demandée est utilisée uniquement pour sélectionner le pool de stockage.

Idéalement, vous pouvez utiliser `attributes` seul pour modéliser les qualités du stockage dont vous avez besoin pour répondre aux besoins d'une classe particulière. Trident découvre et sélectionne automatiquement les pools de stockage correspondant à *all* du `attributes` que vous spécifiez.

Si vous ne parvenez pas à utiliser `attributes` pour sélectionner automatiquement les pools appropriés pour une classe, vous pouvez utiliser les `storagePools` paramètres et `additionalStoragePools` pour affiner davantage les pools ou même pour sélectionner un ensemble spécifique de pools.

Vous pouvez utiliser le `storagePools` paramètre pour restreindre davantage l'ensemble de pools correspondant à n'importe quel `attributes` . En d'autres termes, Trident utilise l'intersection des pools identifiés par les `attributes` paramètres et `storagePools` pour le provisionnement. Vous pouvez utiliser les paramètres seuls ou les deux ensemble.

Vous pouvez utiliser le `additionalStoragePools` paramètre pour étendre l'ensemble des pools utilisés par Trident pour le provisionnement, quels que soient les pools sélectionnés par les `attributes` paramètres et `storagePools`.

Vous pouvez utiliser le `excludeStoragePools` paramètre pour filtrer l'ensemble de pools utilisés par Trident pour le provisionnement. L'utilisation de ce paramètre supprime tous les pools correspondant.

Dans les `storagePools` paramètres et `additionalStoragePools` , chaque entrée prend la forme `<backend>:<storagePoolList>`, où `<storagePoolList>` est une liste de pools de stockage séparés par des virgules pour le back-end spécifié. Par exemple, une valeur pour `additionalStoragePools` peut ressembler à `ontapnas_192.168.1.100:aggr1,aggr2;solidfire_192.168.1.101:bronze`. Ces

listes acceptent les valeurs regex tant pour le back-end que pour les valeurs de liste. Vous pouvez utiliser `tridentctl get backend` pour obtenir la liste des systèmes back-end et leurs pools.

Attributs Kubernetes

Ces attributs n'ont aucun impact sur la sélection des pools de stockage/systèmes back-end par Trident lors du provisionnement dynamique. En effet, ces attributs fournissent simplement les paramètres pris en charge par les volumes persistants de Kubernetes. Les nœuds worker sont responsables des opérations de création de système de fichiers et peuvent nécessiter des utilitaires de système de fichiers, tels que xfsprogs.

Attribut	Type	Valeurs	Description	Facteurs pertinents	Version Kubernetes
Fstype	chaîne	ext4, ext3, xfs	Type de système de fichiers pour les volumes en mode bloc	solidfire-san, ontap-nas, ontap-nas-économie, ontap-nas-flexgroup, ontap-san, ontap-san-économie	Tout
Volumeallowexpansion	booléen	vrai, faux	Activez ou désactivez la prise en charge pour augmenter la taille de la demande de volume persistant	ontap-nas, économie ontap-nas, ontap-nas-flexgroup, ontap-san, ontap-san-économie, solidfire-san, gcp-cvs, azure-netapp-files	1.11+
Volume Bindingmode	chaîne	Immédiat, WaitForFirstConsumer	Sélectionnez le moment où la liaison des volumes et le provisionnement dynamique se produisent	Tout	1.19 - 1.26

- Le `fsType` paramètre permet de contrôler le type de système de fichiers souhaité pour les LUN SAN. En outre, Kubernetes utilise également la présence de `fsType` dans une classe de stockage pour indiquer l'existence d'un système de fichiers. La propriété des volumes peut être contrôlée à l'aide du `fsGroup` contexte de sécurité d'un pod uniquement si `fsType` est défini. Reportez-vous "Kubernetes : configurez un contexte de sécurité pour un pod ou un conteneur"à la pour une vue d'ensemble sur la définition de la propriété du volume à l'aide du `fsGroup` contexte. Kubernetes appliquera la `fsGroup` valeur uniquement si :

  - `fsType` est défini dans la classe de stockage.
  - Le mode d'accès PVC est RWO.

  Pour les pilotes de stockage NFS, un système de fichiers existe déjà dans le cadre de l'exportation NFS. Pour pouvoir utiliser `fsGroup` la classe de stockage, vous devez toujours spécifier une `fsType`. vous pouvez la définir sur `nfs` ou toute valeur non nulle.

- Pour plus d'informations sur l'extension de volume, reportez-vous à la section"Développement des volumes".
- Le programme d'installation de Trident fournit plusieurs exemples de définitions de classe de stockage à utiliser avec Trident dans `sample-input/storage-class-*.yaml`. La suppression d'une classe de stockage Kubernetes entraîne également la suppression de la classe de stockage Trident correspondante.

Objets Kubernetes `VolumeSnapshotClass`

Les objets Kubernetes `VolumeSnapshotClass` sont analogues à `StorageClasses`. Ils aident à définir plusieurs classes de stockage. Ils sont référencés par les snapshots de volume pour associer le snapshot à la classe d'instantanés requise. Chaque snapshot de volume est associé à une classe de snapshot de volume unique.

Un `VolumeSnapshotClass` doit être défini par un administrateur pour créer des instantanés. Une classe de snapshots de volume est créée avec la définition suivante :

```
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshotClass
metadata:
  name: csi-snapclass
driver: csi.trident.netapp.io
deletionPolicy: Delete
```

Le `driver` spécifie à Kubernetes que les demandes de snapshots de volumes de la `csi-snapclass` classe sont gérées par Trident. Le `deletionPolicy` spécifie l'action à effectuer lorsqu'un snapshot doit être supprimé. Lorsque `deletionPolicy` est défini sur `Delete`, les objets de snapshot du volume ainsi que le snapshot sous-jacent du cluster de stockage sont supprimés lors de la suppression d'un snapshot. Vous pouvez également la configurer sur `Retain` signifie que `VolumeSnapshotContent` et le snapshot physique sont conservés.

Objets Kubernetes `VolumeSnapshot`

Un objet Kubernetes `VolumeSnapshot` est une demande de création d'une copie Snapshot d'un volume. Tout comme un volume persistant représente une demande de copie Snapshot d'un volume effectuée par un utilisateur, une copie Snapshot de volume est une demande de création d'un snapshot d'une demande de volume persistant existante.

Lorsqu'une demande de Snapshot de volume est émise, Trident gère automatiquement la création de la copie Snapshot du volume sur le back-end et expose la copie Snapshot en créant un objet unique `VolumeSnapshotContent`. Vous pouvez créer des instantanés à partir de ESV existantes et les utiliser comme source de données lors de la création de nouveaux ESV.

> Le silecyle d'un VolumeSnapshot est indépendant de la demande de volume persistant source : un snapshot persiste même après la suppression de la demande de volume persistant source. Lors de la suppression d'un volume persistant qui possède des snapshots associés, Trident marque le volume de sauvegarde de ce volume persistant dans un état **Suppression**, mais ne le supprime pas complètement. Le volume est supprimé lorsque tous les snapshots associés sont supprimés.

Objets Kubernetes `VolumeSnapshotContent`

Un objet Kubernetes `VolumeSnapshotContent` représente un snapshot d'un volume déjà provisionné. Elle est similaire à un `PersistentVolume` et signifie un snapshot provisionné sur le cluster de stockage. Comme `PersistentVolumeClaim` pour les objets et `PersistentVolume`, lors de la création d'un Snapshot, l' `VolumeSnapshotContent`objet conserve un mappage un-à-un sur l' `VolumeSnapshot`objet qui avait demandé la création du Snapshot.

L' `VolumeSnapshotContent`objet contient des détails qui identifient de manière unique le snapshot, tels que `snapshotHandle`. Il `snapshotHandle` s'agit d'une combinaison unique du nom du PV et du nom de l' `VolumeSnapshotContent`objet.

Lorsqu'une requête de snapshot est fournie, Trident crée le snapshot sur le back-end. Une fois le snapshot créé, Trident configure un `VolumeSnapshotContent` objet et expose ce dernier à l'API Kubernetes.

> En général, il n'est pas nécessaire de gérer `VolumeSnapshotContent` l'objet. Une exception à cette règle s'applique lorsque vous souhaitez "importer un instantané de volume"créer des éléments en dehors de Trident.

Objets Kubernetes `CustomResourceDefinition`

Les ressources personnalisées Kubernetes sont des terminaux de l'API Kubernetes définis par l'administrateur et utilisés pour regrouper des objets similaires. Kubernetes prend en charge la création de ressources personnalisées pour le stockage d'une collection d'objets. Vous pouvez obtenir ces définitions de ressources en exécutant `kubectl get crds`.

Les définitions de ressources personnalisées (CRD) et les métadonnées d'objet associées sont stockées sur le magasin de métadonnées Kubernetes. Ce qui évite d'avoir recours à un magasin séparé pour Trident.

Trident utilise `CustomResourceDefinition` des objets pour préserver l'identité des objets Trident, tels que les systèmes back-end Trident, les classes de stockage Trident et les volumes Trident. Ces objets sont gérés par Trident. En outre, la structure d'instantané de volume CSI introduit quelques CRD nécessaires pour définir des instantanés de volume.

Les CRDS sont une construction Kubernetes. Les objets des ressources définies ci-dessus sont créés par Trident. Par exemple simple, lorsqu'un back-end est créé à l'aide de `tridentctl`, un objet CRD correspondant `tridentbackends` est créé pour être consommé par Kubernetes.

Voici quelques points à garder à l'esprit sur les CRD de Trident :

- Lorsque Trident est installé, un ensemble de CRD est créé et peut être utilisé comme tout autre type de ressource.
- Lors de la désinstallation de Trident à l'aide de la `tridentctl uninstall` commande, les modules Trident sont supprimés, mais les CRD créés ne sont pas nettoyés. Reportez-vous "Désinstaller Trident"à la pour comprendre comment Trident peut être complètement supprimé et reconfiguré de zéro.

ObjetsTrident `StorageClass`

Trident crée des classes de stockage correspondantes pour les objets Kubernetes `StorageClass` qui spécifient `csi.trident.netapp.io` dans leur champ de provisionnement. Le nom de classe de stockage correspond à celui de l'objet Kubernetes `StorageClass` qu'il représente.

> Avec Kubernetes, ces objets sont créés automatiquement lorsqu'un Kubernetes `StorageClass` qui utilise Trident en tant que provisionneur est enregistré.

Les classes de stockage comprennent un ensemble d'exigences pour les volumes. Trident mappe ces exigences avec les attributs présents dans chaque pool de stockage. S'ils correspondent, ce pool de stockage est une cible valide pour le provisionnement des volumes qui utilisent cette classe de stockage.

Vous pouvez créer des configurations de classes de stockage afin de définir directement des classes de stockage à l'aide de l'API REST. Toutefois, pour les déploiements Kubernetes, nous prévoyons qu'ils seront créés lors de l'enregistrement de nouveaux objets Kubernetes `StorageClass`.

Objets back-end Trident

Les systèmes back-end représentent les fournisseurs de stockage au-dessus desquels Trident provisionne des volumes. Une instance Trident unique peut gérer un nombre illimité de systèmes back-end.

> Il s'agit de l'un des deux types d'objet que vous créez et gérez vous-même. L'autre est l'objet Kubernetes `StorageClass`.

Pour plus d'informations sur la construction de ces objets, reportez-vous à la section "configuration des systèmes back-end".

ObjetsTrident `StoragePool`

Les pools de stockage représentent les emplacements distincts disponibles pour le provisionnement sur chaque système back-end. Pour ONTAP, ces derniers correspondent à des agrégats dans des SVM. Pour NetApp HCI/SolidFire, ils correspondent aux bandes QoS spécifiées par l'administrateur. Pour Cloud Volumes Service, ces régions correspondent à des régions du fournisseur cloud. Chaque pool de stockage dispose d'un ensemble d'attributs de stockage distincts, qui définissent ses caractéristiques de performances et ses caractéristiques de protection des données.

Contrairement aux autres objets ici, les candidats au pool de stockage sont toujours découverts et gérés automatiquement.

ObjetsTrident `Volume`

Les volumes constituent l'unité de provisionnement de base, comprenant des terminaux back-end, tels que des partages NFS et des LUN iSCSI et FC. Dans Kubernetes, ces valeurs correspondent directement à `PersistentVolumes`. Lorsque vous créez un volume, assurez-vous qu'il possède une classe de stockage, qui détermine l'emplacement de provisionnement de ce volume, ainsi que sa taille.

- Dans Kubernetes, ces objets sont gérés automatiquement. Vous pouvez les afficher pour voir le provisionnement Trident.
- Lors de la suppression d'un volume persistant avec des snapshots associés, le volume Trident correspondant est mis à jour avec un état **Suppression**. Pour que le volume Trident soit supprimé, vous devez supprimer les snapshots du volume.

Une configuration de volume définit les propriétés qu'un volume provisionné doit avoir.

Attribut	Type	Obligatoire	Description
version	chaîne	non	Version de l'API Trident (« 1 »)
nom	chaîne	oui	Nom du volume à créer
Classe de stockage	chaîne	oui	Classe de stockage à utiliser lors du provisionnement du volume
taille	chaîne	oui	Taille du volume à provisionner en octets
protocole	chaîne	non	Type de protocole à utiliser : « fichier » ou « bloc »
Nom interne	chaîne	non	Nom de l'objet sur le système de stockage, généré par Trident
Volume cloneSourceVolume	chaîne	non	ONTAP (nas, san) et SolidFire-* : nom du volume à cloner
SplitOnClone	chaîne	non	ONTAP (nas, san) : séparer le clone de son parent
Politique de snapshots	chaîne	non	ONTAP-* : stratégie d'instantané à utiliser
Réserve de snapshots	chaîne	non	ONTAP-* : pourcentage de volume réservé pour les snapshots
ExportPolicy	chaîne	non	ontap-nas* : export policy à utiliser

Attribut	Type	Obligatoire	Description
Répertoire de snapshots	bool	non	ontap-nas* : indique si le répertoire des snapshots est visible
Autorisations unix	chaîne	non	ontap-nas* : autorisations UNIX initiales
Taille de bloc	chaîne	non	SolidFire-*: Taille de bloc/secteur
Système de fichiers	chaîne	non	Type de système de fichiers

Trident génère `internalName` lors de la création du volume. Il s'agit de deux étapes. Tout d'abord, il ajoute le préfixe de stockage (le préfixe par défaut `trident` ou le préfixe dans la configuration back-end) au nom du volume, ce qui entraîne un nom de formulaire `<prefix>-<volume-name>`. Il procède ensuite à la désinfection du nom en remplaçant les caractères non autorisés dans le back-end. Pour les systèmes ONTAP back-end, il remplace les tirets par des traits de soulignement (le nom interne devient ainsi `<prefix>_<volume-name>`). Pour les systèmes back-end Element, il remplace les tirets de traits de soulignement.

Vous pouvez utiliser des configurations de volume pour provisionner directement des volumes à l'aide de l'API REST, mais dans les déploiements Kubernetes, la plupart des utilisateurs doivent utiliser la méthode Kubernetes standard `PersistentVolumeClaim`. Trident crée automatiquement cet objet volume dans le cadre du provisionnement.

ObjetsTrident `Snapshot`

Les snapshots sont une copie de volumes à un point dans le temps, qui peut être utilisée pour provisionner de nouveaux volumes ou restaurer l'état de ces volumes. Dans Kubernetes, ces données correspondent directement aux `VolumeSnapshotContent` objets. Chaque snapshot est associé à un volume, qui est la source des données du snapshot.

Chaque `Snapshot` objet comprend les propriétés répertoriées ci-dessous :

Attribut	Type	Obligatoire	Description
version	Chaîne	Oui	Version de l'API Trident (« 1 »)
nom	Chaîne	Oui	Nom de l'objet snapshot Trident
Nom interne	Chaîne	Oui	Nom de l'objet Snapshot Trident sur le système de stockage
Nom du volume	Chaîne	Oui	Nom du volume persistant pour lequel le snapshot est créé
Volume Nom interne	Chaîne	Oui	Nom de l'objet volume Trident associé sur le système de stockage

Dans Kubernetes, ces objets sont gérés automatiquement. Vous pouvez les afficher pour voir le provisionnement Trident.

Lors de la création d'une demande d'objet Kubernetes `VolumeSnapshot`, Trident crée un objet de snapshot sur le système de stockage qui soutient. Le `internalName` de cet objet de snapshot est généré en combinant le préfixe `snapshot-` et le `UID` de l' `VolumeSnapshot`objet (par exemple, `snapshot-e8d8a0ca-9826-11e9-9807-525400f3f660`). `volumeName` et `volumeInternalName` sont renseignés en obtenant les détails du volume de sauvegarde.

ObjetTrident `ResourceQuota`

La déamonset Trident consomme une `system-node-critical` classe de priorité, la classe de priorité la plus élevée disponible dans Kubernetes, pour s'assurer que Trident peut identifier et nettoyer les volumes lors de l'arrêt normal des nœuds et permettre aux pods de diaboset Trident d'anticiper les charges de travail avec une priorité inférieure dans les clusters où la pression de ressources est élevée.

Pour ce faire, Trident utilise un `ResourceQuota` objet afin de s'assurer qu'une classe de priorité « système-noeud-critique » sur le démonset Trident est satisfaite. Avant le déploiement et la création de démonset, Trident recherche l' `ResourceQuota`objet et, s'il n'est pas découvert, l'applique.

Si vous avez besoin de plus de contrôle sur le quota de ressources par défaut et la classe de priorité, vous pouvez générer un `custom.yaml` ou configurer l' `ResourceQuota`objet à l'aide du graphique Helm.

Voici un exemple de `Resourcequota"objet hiérarchisant le demonset Trident.

```
apiVersion: <version>
kind: ResourceQuota
metadata:
  name: trident-csi
  labels:
    app: node.csi.trident.netapp.io
spec:
  scopeSelector:
    matchExpressions:
      - operator: In
        scopeName: PriorityClass
        values:
          - system-node-critical
```

Pour plus d'informations sur les quotas de ressources, reportez-vous "Kubernetes : quotas de ressources"à la section .

Nettoyez `ResourceQuota` si l'installation échoue

Dans les rares cas où l'installation échoue après la `ResourceQuota` création de l'objet, essayez d'abord, "désinstallation"puis réinstallez.

Si cela ne fonctionne pas, supprimez manuellement l' `ResourceQuota`objet.

Déposer `ResourceQuota`

Si vous préférez contrôler votre propre allocation de ressources, vous pouvez supprimer l'objet Trident `ResourceQuota` à l'aide de la commande :

```
kubectl delete quota trident-csi -n trident
```

Normes de sécurité de pod (PSS) et contraintes de contexte de sécurité (SCC)

Les normes de sécurité de Kubernetes Pod (PSS) et les règles de sécurité de Pod (PSP) définissent des niveaux d'autorisation et limitent le comportement des pods. OpenShift Security Context Constraints (SCC) définit de façon similaire les restrictions de pod spécifiques à OpenShift Kubernetes Engine. Pour fournir cette personnalisation, Trident active certaines autorisations pendant l'installation. Les sections suivantes détaillent les autorisations définies par Trident.

PSS remplace les politiques de sécurité Pod (PSP). La PSP est obsolète dans Kubernetes v1.21 et elle sera supprimée dans la version 1.25. Pour plus d'informations, reportez-vous "Kubernetes : sécurité"à .

Contexte de sécurité Kubernetes requis et champs associés

Autorisations	Description
Privilégié	CSI nécessite que les points de montage soient bidirectionnels, ce qui signifie que le pod de nœud Trident doit exécuter un conteneur privilégié. Pour plus d'informations, reportez-vous "Kubernetes : propagation du montage"à .
Mise en réseau d'hôtes	Requis pour le démon iSCSI. `iscsiadm` Gère les montages iSCSI et utilise la mise en réseau des hôtes pour communiquer avec le démon iSCSI.
IPC de l'hôte	Le NFS utilise la communication interprocess (IPC) pour communiquer avec le NFSD.
PID de l'hôte	Nécessaire pour démarrer `rpc-statd` pour NFS. Trident interroge les processus hôtes pour déterminer si `rpc-statd` est en cours d'exécution avant le montage des volumes NFS.

Capacités	Cette `SYS_ADMIN` fonctionnalité est fournie dans le cadre des fonctionnalités par défaut pour les conteneurs privilégiés. Par exemple, Docker définit les fonctionnalités suivantes pour les conteneurs privilégiés :
	`CapPrm: 0000003fffffffff`
	`CapEff: 0000003fffffffff`

Autorisations	Description
Seccomp	Le profil Seccomp est toujours « non confiné » dans des conteneurs privilégiés ; par conséquent, il ne peut pas être activé dans Trident.
SELinux	Sur OpenShift, les conteneurs privilégiés sont exécutés dans `spc_t` le domaine (« conteneur super privilégié ») et les conteneurs non privilégiés dans le `container_t` domaine. Sur `containerd`, avec `container-selinux` installé, tous les conteneurs sont exécutés dans le `spc_t` domaine, ce qui désactive effectivement SELinux. Par conséquent, Trident n'ajoute pas `seLinuxOptions` aux conteneurs.
DAC	Les conteneurs privilégiés doivent être exécutés en tant que root. Les conteneurs non privilégiés s'exécutent comme root pour accéder aux sockets unix requis par CSI.

Normes de sécurité du pod (PSS)

Étiquette	Description	Valeur par défaut
`pod-security.kubernetes.io/enforce` `pod-security.kubernetes.io/enforce-version`	Permet au contrôleur et aux nœuds Trident d'être admis dans le namespace d'installation. Ne modifiez pas le libellé de l'espace de noms.	`enforce: privileged enforce-version: <version of the current cluster or highest version of PSS tested.>`

La modification des étiquettes de l'espace de noms peut entraîner l'absence de planification des modules, un "erreur de création: …" ou un "avertissement: trident-csi-…". Dans ce cas, vérifiez si le libellé de l'espace de noms pour `privileged` a été modifié. Si c'est le cas, réinstallez Trident.

Politiques de sécurité des pods (PSP)

Champ	Description	Valeur par défaut
`allowPrivilegeEscalation`	Les conteneurs privilégiés doivent autoriser l'escalade des privilèges.	`true`
`allowedCSIDrivers`	Trident n'utilise pas les volumes éphémères CSI en ligne.	Vide
`allowedCapabilities`	Les conteneurs Trident non privilégiés ne nécessitent pas de fonctionnalités supérieures à celles des ensembles par défaut et les conteneurs privilégiés se voient accorder toutes les capacités possibles.	Vide

Champ	Description	Valeur par défaut
`allowedFlexVolumes`	Trident n'utilise pas un "Pilote FlexVolume", donc ils ne sont pas inclus dans la liste des volumes autorisés.	Vide
`allowedHostPaths`	Le pod des nœuds Trident monte le système de fichiers racine du nœud, ce qui ne permet donc pas de définir cette liste.	Vide
`allowedProcMountTypes`	Trident n'utilise aucun `ProcMountTypes`.	Vide
`allowedUnsafeSysctls`	Trident n'exige pas de sécurité `sysctls`.	Vide
`defaultAddCapabilities`	Aucune fonctionnalité n'est requise pour être ajoutée aux conteneurs privilégiés.	Vide
`defaultAllowPrivilegeEscalation`	L'autorisation de réaffectation des privilèges est gérée dans chaque pod Trident.	`false`
`forbiddenSysctls`	Non `sysctls` sont autorisés.	Vide
`fsGroup`	Les conteneurs Trident s'exécutent en tant que root.	`RunAsAny`
`hostIPC`	Le montage des volumes NFS nécessite la communication de l'IPC hôte avec `nfsd`	`true`
`hostNetwork`	Iscsiadm nécessite que le réseau hôte communique avec le démon iSCSI.	`true`
`hostPID`	Le PID de l'hôte est nécessaire pour vérifier si `rpc-statd` est exécuté sur le nœud.	`true`
`hostPorts`	Trident n'utilise aucun port hôte.	Vide
`privileged`	Les pods de nœuds Trident doivent exécuter un conteneur privilégié pour monter des volumes.	`true`
`readOnlyRootFilesystem`	Les pods de nœuds Trident doivent écrire dans le système de fichiers de nœuds.	`false`
`requiredDropCapabilities`	Les pods de nœuds Trident exécutent un conteneur privilégié et ne peuvent pas supprimer de fonctionnalités.	`none`
`runAsGroup`	Les conteneurs Trident s'exécutent en tant que root.	`RunAsAny`

Champ	Description	Valeur par défaut
`runAsUser`	Les conteneurs Trident s'exécutent en tant que root.	`runAsAny`
`runtimeClass`	Trident n'utilise pas `RuntimeClasses` .	Vide
`seLinux`	Trident ne définit pas `seLinuxOptions`, car il existe actuellement des différences dans la façon dont les temps d'exécution des conteneurs et les distributions Kubernetes gèrent SELinux.	Vide
`supplementalGroups`	Les conteneurs Trident s'exécutent en tant que root.	`RunAsAny`
`volumes`	Les pods Trident requièrent ces plug-ins de volume.	`hostPath, projected, emptyDir`

Contraintes de contexte de sécurité (SCC)

Étiquettes	Description	Valeur par défaut
`allowHostDirVolumePlugin`	Les pods des nœuds Trident montent le système de fichiers racine du nœud.	`true`
`allowHostIPC`	Le montage des volumes NFS nécessite la communication de l'IPC hôte avec `nfsd`.	`true`
`allowHostNetwork`	Iscsiadm nécessite que le réseau hôte communique avec le démon iSCSI.	`true`
`allowHostPID`	Le PID de l'hôte est nécessaire pour vérifier si `rpc-statd` est exécuté sur le nœud.	`true`
`allowHostPorts`	Trident n'utilise aucun port hôte.	`false`
`allowPrivilegeEscalation`	Les conteneurs privilégiés doivent autoriser l'escalade des privilèges.	`true`
`allowPrivilegedContainer`	Les pods de nœuds Trident doivent exécuter un conteneur privilégié pour monter des volumes.	`true`
`allowedUnsafeSysctls`	Trident n'exige pas de sécurité `sysctls`.	`none`

Étiquettes	Description	Valeur par défaut
`allowedCapabilities`	Les conteneurs Trident non privilégiés ne nécessitent pas de fonctionnalités supérieures à celles des ensembles par défaut et les conteneurs privilégiés se voient accorder toutes les capacités possibles.	Vide
`defaultAddCapabilities`	Aucune fonctionnalité n'est requise pour être ajoutée aux conteneurs privilégiés.	Vide
`fsGroup`	Les conteneurs Trident s'exécutent en tant que root.	`RunAsAny`
`groups`	Ce SCC est spécifique à Trident et lié à son utilisateur.	Vide
`readOnlyRootFilesystem`	Les pods de nœuds Trident doivent écrire dans le système de fichiers de nœuds.	`false`
`requiredDropCapabilities`	Les pods de nœuds Trident exécutent un conteneur privilégié et ne peuvent pas supprimer de fonctionnalités.	`none`
`runAsUser`	Les conteneurs Trident s'exécutent en tant que root.	`RunAsAny`
`seLinuxContext`	Trident ne définit pas `seLinuxOptions`, car il existe actuellement des différences dans la façon dont les temps d'exécution des conteneurs et les distributions Kubernetes gèrent SELinux.	Vide
`seccompProfiles`	Les conteneurs privilégiés s'exécutent toujours « sans limite ».	Vide
`supplementalGroups`	Les conteneurs Trident s'exécutent en tant que root.	`RunAsAny`
`users`	Une entrée est fournie pour lier ce SCC à l'utilisateur Trident dans l'espace de noms Trident.	s/o
`volumes`	Les pods Trident requièrent ces plug-ins de volume.	`hostPath, downwardAPI, projected, emptyDir`

Informations sur le copyright

Informations sur les marques commerciales