Référence

Trident

NetApp
January 15, 2026

Sommaire

Référence . . . 1
- Ports Trident . . . 1
  - Ports Trident . . . 1
- API REST Trident . . . 1
  - Quand utiliser l'API REST . . . 1
  - Utilisation de l'API REST . . . 1
- Options de ligne de commande . . . 2
  - Enregistrement . . . 2
  - Kubernetes . . . 2
  - Docker . . . 3
  - REPOS . . . 3
- Objets Kubernetes et Trident . . . 3
  - Comment les objets interagissent-ils entre eux ? . . . 3
  - Kubernetes `PersistentVolumeClaim` objets . . . 4
  - Kubernetes `PersistentVolume` objets . . . 5
  - Kubernetes `StorageClass` objets . . . 6
  - Kubernetes `VolumeSnapshotClass` objets . . . 10
  - Kubernetes `VolumeSnapshot` objets . . . 10
  - Kubernetes `VolumeSnapshotContent` objets . . . 11
  - Kubernetes `VolumeGroupSnapshotClass` objets . . . 11
  - Kubernetes `VolumeGroupSnapshot` objets . . . 12
  - Kubernetes `VolumeGroupSnapshotContent` objets . . . 12
  - Kubernetes `CustomResourceDefinition` objets . . . 13
  - Trident `StorageClass` objets . . . 13
  - Objets backend Trident . . . 13
  - Trident `StoragePool` objets . . . 14
  - Trident `Volume` objets . . . 14
  - Trident `Snapshot` objets . . . 15
  - Trident `ResourceQuota` objet . . . 16
- Normes de sécurité des modules (PSS) et contraintes de contexte de sécurité (SCC) . . . 17
  - Contexte de sécurité Kubernetes requis et champs associés . . . 18
  - Normes de sécurité des capsules (PSS) . . . 18
  - Politiques de sécurité des pods (PSP) . . . 19
  - Contraintes de contexte de sécurité (CCS) . . . 20

Référence

Ports Trident

Découvrez plus d'informations sur les ports utilisés par Trident pour la communication.

Ports Trident

Trident utilise les ports suivants pour la communication au sein de Kubernetes :

Port	But
8443	HTTPS de canal de retour
8001	point de terminaison des métriques Prometheus
8000	Serveur REST Trident
17546	Port de sonde de disponibilité/d'état de fonctionnement utilisé par les pods du démon Trident

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

API REST Trident

Alors que"commandes et options de tridentctl" sont le moyen le plus simple d'interagir avec l'API REST de Trident , mais vous pouvez utiliser directement le point de terminaison REST si vous préférez.

Quand utiliser l'API REST

L'API REST est utile pour les installations avancées qui utilisent Trident comme binaire autonome dans des déploiements non-Kubernetes.

Pour une meilleure sécurité, le Trident `REST API` par défaut, l'exécution dans un pod est limitée à l'hôte local. Pour modifier ce comportement, vous devez configurer les paramètres de Trident. `-address` argument dans sa configuration de pod.

Utilisation de l'API REST

Pour des exemples d'appels à ces API, transmettez le débogage(`-d`) drapeau. Pour plus d'informations, veuillez consulter"Gérez Trident à l'aide de tridentctl" .

L'API fonctionne comme suit :

OBTENIR

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

Liste tous les objets de ce type.

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

Obtient les détails de l'objet nommé.

POSTE

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

Crée un objet du type spécifié.

- Nécessite une configuration JSON pour que l'objet soit créé. Pour connaître les spécifications de chaque type d'objet, veuillez vous référer à"Gérez Trident à l'aide de tridentctl" .
- Si l'objet existe déjà, le comportement varie : les backends mettent à jour l'objet existant, tandis que tous les autres types d'objets échoueront lors de l'opération.

SUPPRIMER

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

Supprime la ressource nommée.

> Les volumes associés aux serveurs backend ou aux classes de stockage continueront d'exister ; ceux-ci doivent être supprimés séparément. Pour plus d'informations, veuillez consulter"Gérez Trident à l'aide de tridentctl" .

Options de ligne de commande

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

Enregistrement

`-debug`

Active la sortie de débogage.

`-loglevel <level>`

Définit le niveau de journalisation (débogage, info, avertissement, erreur, fatal). Valeur par défaut : info.

Kubernetes

`-k8s_pod`

Utilisez cette option ou `-k8s_api_server` pour activer la prise en charge de Kubernetes. Ce paramétrage permet à Trident d'utiliser les informations d'identification du compte de service Kubernetes du pod conteneur pour contacter le serveur API. Cela ne fonctionne que lorsque Trident est exécuté en tant que pod dans un cluster Kubernetes avec des 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. Lorsque cela est spécifié, Trident se connecte au serveur d'API Kubernetes en utilisant l'adresse et le port non sécurisés fournis. Cela

permet de déployer Trident en dehors d'un pod ; cependant, cela ne prend en charge que les connexions non sécurisées au serveur API. Pour une connexion sécurisée, déployez Trident dans un pod avec le `-k8s_pod` option.

Docker

`-volume_driver <name>`

Nom du pilote utilisé lors de l'enregistrement du plugin Docker. Valeur par défaut `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 d'accès à un fichier de configuration backend.

REPOS

`-address <ip-or-host>`

Spécifie l'adresse sur laquelle le serveur REST de Trident doit écouter. Par défaut, localhost. Lorsqu'elle est exécutée en local et dans un pod Kubernetes, l'interface REST n'est pas directement accessible depuis l'extérieur du pod. Utiliser `-address ""` rendre l'interface REST accessible depuis l'adresse IP du pod.

L'interface REST de Trident peut être configurée pour écouter et servir uniquement à l'adresse 127.0.0.1 (pour IPv4) ou [::1] (pour IPv6).

`-port <port-number>`

Spécifie le port sur lequel le serveur REST de Trident doit écouter. Par défaut, 8000.

`-rest`

Active l'interface REST. Par défaut, la valeur est « vrai ».

Objets Kubernetes et Trident

Vous pouvez interagir avec Kubernetes et Trident à l'aide d'API REST en lisant et en écrivant des objets de ressources. Plusieurs objets de ressources déterminent la relation entre Kubernetes et Trident, Trident et le stockage, et Kubernetes et le stockage. Certains de ces objets sont gérés via Kubernetes et les autres via Trident.

Comment les objets interagissent-ils entre eux ?

La manière la plus simple de comprendre ces objets, leur utilité et leurs interactions consiste peut-être à suivre une simple requête de stockage provenant d'un utilisateur Kubernetes :

1. Un utilisateur crée un `PersistentVolumeClaim` demander un nouveau `PersistentVolume` d'une taille particulière à partir d'un Kubernetes `StorageClass` qui avait été configurée précédemment par l'administrateur.
2. Kubernetes `StorageClass` identifie Trident comme son fournisseur et inclut des paramètres qui indiquent à Trident comment provisionner un volume pour la classe demandée.

3. Trident examine sa propre `StorageClass` portant le 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 backend correspondant et crée deux objets : un `PersistentVolume` dans Kubernetes qui indique à Kubernetes comment trouver, monter et traiter le volume, et un volume dans Trident qui conserve la relation entre les `PersistentVolume` et le stockage proprement dit.
5. Kubernetes lie les `PersistentVolumeClaim` au nouveau `PersistentVolume` . Des capsules qui comprennent les `PersistentVolumeClaim` Montez ce PersistentVolume sur n'importe quel hôte sur lequel il s'exécute.
6. Un utilisateur crée un `VolumeSnapshot` d'un PVC existant, en utilisant un `VolumeSnapshotClass` cela désigne Trident.
7. Trident identifie le volume associé au PVC et crée un instantané de ce volume sur son système backend. Cela crée également un `VolumeSnapshotContent` qui indique à Kubernetes comment identifier l'instantané.
8. Un utilisateur peut créer un `PersistentVolumeClaim` en utilisant `VolumeSnapshot` comme source.
9. Trident identifie l'instantané requis et effectue la même série d'étapes que pour la création d'un `PersistentVolume` et un `Volume` .

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

Kubernetes `PersistentVolumeClaim` objets

Un Kubernetes `PersistentVolumeClaim` L'objet représente une demande de stockage effectuée par un utilisateur d'un cluster Kubernetes.

En plus de la spécification standard, Trident permet aux utilisateurs de spécifier les annotations spécifiques au volume suivantes s'ils souhaitent remplacer les valeurs par défaut définies dans la configuration du backend :

Annotation	Option de volume	Pilotes pris en charge
trident.netapp.io/fileSystem	système de fichiers	ontap-san, solidfire-san, ontap-san-économie
trident.netapp.io/cloneFromPVC	cloneSourceVolume	ontap-nas, ontap-san, solidfire-san, azure-netapp-files, gcp-cvs, ontap-san-economy
trident.netapp.io/splitOnClone	splitOnClone	ontap-nas, ontap-san
trident.netapp.io/protocol	protocole	n'importe lequel
trident.netapp.io/exportPolicy	politique d'exportation	ontap-nas, ontap-nas-économie, ontap-nas-flexgroup
trident.netapp.io/snapshotPolicy	instantanéPolitique	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san
trident.netapp.io/snapshotReserve	snapshotReserve	ontap-nas, ontap-nas-flexgroup, ontap-san, gcp-cvs
trident.netapp.io/snapshotDirectory	répertoire snapshot	ontap-nas, ontap-nas-économie, ontap-nas-flexgroup

Annotation	Option de volume	Pilotes pris en charge
trident.netapp.io/unixPermissions	Autorisations Unix	ontap-nas, ontap-nas-économie, ontap-nas-flexgroup
trident.netapp.io/blockSize	taille du bloc	solidefire-san

Si le PV créé a le `Delete` En vertu de la politique de récupération, Trident supprime à la fois le PV et le volume de sauvegarde lorsque le PV est libéré (c'est-à-dire lorsque l'utilisateur supprime le PVC). En cas d'échec de la suppression, Trident marque le PV comme tel et réessaie périodiquement l'opération jusqu'à ce qu'elle réussisse ou que le PV soit supprimé manuellement. Si le PV utilise le `Retain` Trident ignore cette politique et suppose que l'administrateur la supprimera de Kubernetes et du système dorsal, permettant ainsi de sauvegarder ou d'inspecter le volume avant sa suppression. Notez que la suppression du PV n'entraîne pas la suppression du volume de sauvegarde par Trident . Vous devriez le supprimer à l'aide de l'API REST(`tridentctl`).

Trident prend en charge la création d'instantanés de volume à l'aide de la spécification CSI : vous pouvez créer un instantané de volume et l'utiliser comme source de données pour cloner des PVC existants. De cette manière, des copies ponctuelles des PV peuvent être exposées à Kubernetes sous forme d'instantanés. Ces instantanés peuvent ensuite être utilisés pour créer de nouveaux volumes persistants. Jetez un oeil à `On-Demand Volume Snapshots` pour voir comment cela fonctionnerait.

Trident fournit également `cloneFromPVC` et `splitOnClone` annotations pour la création de clones. Vous pouvez utiliser ces annotations pour cloner un PVC sans avoir à utiliser l'implémentation CSI.

Voici un exemple : si un utilisateur possède déjà un PVC appelé `mysql` L'utilisateur peut créer un nouveau PVC appelé `mysqlclone` en utilisant l'annotation, par exemple `trident.netapp.io/cloneFromPVC: mysql` . Avec cet ensemble d'annotations, Trident clone le volume correspondant au PVC mysql, au lieu de provisionner un volume à partir de zéro.

Considérez les points suivants :

- NetApp recommande de cloner un volume inactif.
- Un PVC et son clone doivent se trouver dans le même espace de noms Kubernetes et avoir la même classe de stockage.
- Avec le `ontap-nas` et `ontap-san` Pour les conducteurs, il pourrait être souhaitable de définir l'annotation PVC `trident.netapp.io/splitOnClone` en conjonction avec `trident.netapp.io/cloneFromPVC` . Avec `trident.netapp.io/splitOnClone` défini à `true` Trident sépare le volume cloné du volume parent et découple ainsi complètement le cycle de vie du volume cloné de celui de son parent, au prix d'une perte d'efficacité de stockage. Ne pas paramétrer `trident.netapp.io/splitOnClone` ou en le paramétrant `false` Il en résulte une réduction de la consommation d'espace sur le système dorsal, au prix de la création de dépendances entre les volumes parent et clone, de sorte que le volume parent ne peut être supprimé que si le clone est supprimé au préalable. Un scénario où la division du clone est judicieuse est le clonage d'un volume de base de données vide, où l'on s'attend à ce que le volume et son clone divergent considérablement et ne bénéficient pas des gains d'efficacité de stockage offerts par ONTAP.

Le `sample-input` Ce répertoire contient des exemples de définitions PVC à utiliser avec Trident. Se référer à pour une description complète des paramètres et réglages associés aux volumes Trident .

Kubernetes `PersistentVolume` objets

Un Kubernetes `PersistentVolume` L'objet représente un élément de stockage mis à la disposition du cluster

Kubernetes. Son cycle de vie est indépendant du pod qui l'utilise.

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

Lorsque vous créez un PVC qui fait référence à un Trident-based `StorageClass` Trident provisionne un nouveau volume en utilisant la classe de stockage correspondante et enregistre un nouveau PV pour ce volume. Lors de la configuration du volume provisionné et du PV correspondant, Trident suit les règles suivantes :

- Trident génère un nom de PV pour Kubernetes et un nom interne qu'il utilise pour provisionner le stockage. Dans les deux cas, cela garantit que les noms sont uniques dans leur portée.
- Le volume correspond au mieux à la taille demandée dans le PVC, bien qu'il puisse être arrondi à la quantité allouable la plus proche, selon la plateforme.

Kubernetes `StorageClass` objets

Kubernetes `StorageClass` Les objets sont spécifiés par leur nom dans `PersistentVolumeClaims` provisionner un espace de stockage avec un ensemble de propriétés. La classe de stockage elle-même identifie le fournisseur à utiliser et définit cet ensemble de propriétés en des termes que le fournisseur comprend.

Il s'agit de l'un des deux objets de base que l'administrateur doit créer et gérer. L'autre est l'objet backend Trident .

Un Kubernetes `StorageClass` Un objet utilisant Trident ressemble à ceci :

```
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 les volumes pour cette classe.

Les paramètres de la classe de stockage sont :

Attribut	Type	Obligatoire	Description
attributs	carte[chaîne]chaîne	Non	Voir la section attributs ci-dessous

Attribut	Type	Obligatoire	Description
Piscines de stockage	map[chaîne]StringList	Non	Carte des noms de backend vers des listes de pools de stockage au sein
Bassins de stockage supplémentaires	map[chaîne]StringList	Non	Carte des noms de backend vers des listes de pools de stockage au sein
exclure les pools de stockage	map[chaîne]StringList	Non	Carte des noms de backend vers des listes de pools de stockage au sein

Les attributs de stockage et leurs valeurs possibles peuvent être classés en attributs de sélection du pool 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 des volumes d'un type donné.

Attribut	Type	Valeurs	Offre	Demande	Soutenu par
médias[1]	chaîne	disque dur, hybride, SSD	La piscine contient des médias de ce type ; hybride signifie à la fois	Type de média spécifié	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, solidfire-san
type de provisionnement	chaîne	mince, épais	Pool prend en charge cette méthode d'approvisionnement	Méthode de provisionnement spécifiée	Épais : tous les produits Ontap ; mince : tous les produits Ontap et Solidfire-San
Type de backend	chaîne	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, solidfire-san, gcp-cvs, azure-netapp-files, ontap-san-economy	Pool appartient à ce type de backend	Backend spécifié	Tous les conducteurs
instantanés	booléen	vrai, faux	Pool prend en charge les volumes avec instantanés	Volume avec instantanés activés	ontap-nas, ontap-san, solidfire-san, gcp-cvs

Attribut	Type	Valeurs	Offre	Demande	Soutenu par
clones	booléen	vrai, faux	Pool prend en charge les volumes de clonage	Volume avec clones activés	ontap-nas, ontap-san, solidfire-san, gcp-cvs
cryptage	booléen	vrai, faux	Pool prend en charge les volumes chiffrés	Volume avec chiffrement activé	ontap-nas, ontap-nas-économie, ontap-nas-groupes flexibles, ontap-san
Op E/S par sec	int	entier positif	Pool est capable de garantir des IOPS dans cette plage.	Volume garanti pour ces IOPS	solidefire-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, demander un provisionnement épais entraîne la création d'un volume provisionné de manière épaisse. Cependant, un pool de stockage Element utilise ses valeurs IOPS minimales et maximales offertes pour définir les valeurs QoS, plutôt que la valeur demandée. Dans ce cas, la valeur demandée sert uniquement à sélectionner le pool de stockage.

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

Si vous vous trouvez dans l'incapacité d'utiliser `attributes` Pour sélectionner automatiquement les bons pools pour une classe, vous pouvez utiliser le `storagePools` et `additionalStoragePools` des paramètres permettant d'affiner davantage les pools, voire de sélectionner un ensemble spécifique de pools.

Vous pouvez utiliser le `storagePools` paramètre permettant de restreindre davantage l'ensemble des pools correspondant à une spécification donnée `attributes` . Autrement dit, Trident utilise l'intersection des bassins identifiés par le `attributes` et `storagePools` paramètres de provisionnement. Vous pouvez utiliser chaque paramètre seul ou les deux ensemble.

Vous pouvez utiliser le `additionalStoragePools` paramètre permettant d'étendre l'ensemble des pools que Trident utilise pour le provisionnement, indépendamment des pools sélectionnés par le `attributes` et `storagePools` paramètres.

Vous pouvez utiliser le `excludeStoragePools` Paramètre permettant de filtrer l'ensemble des pools que Trident utilise pour le provisionnement. L'utilisation de ce paramètre supprime tous les pools correspondants.

Dans le `storagePools` et `additionalStoragePools` en paramètres, chaque entrée prend la forme `<backend>:<storagePoolList>` , où `<storagePoolList>` est une liste de pools de stockage séparés par des virgules pour le backend spécifié. Par exemple, une valeur pour `additionalStoragePools` pourrait ressembler à `ontapnas_192.168.1.100:aggr1,aggr2;solidfire_192.168.1.101:bronze` . Ces listes acceptent les valeurs regex pour le backend et les valeurs de la liste. Vous pouvez utiliser `tridentctl get backend` pour obtenir la liste des serveurs backend et de leurs pools.

attributs Kubernetes

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

Attribut	Type	Valeurs	Description	Conducteurs pertinents	Version Kubernetes
fsType	chaîne	ext4, ext3, xfs	Le type de système de fichiers pour les volumes de blocs	solidfire-san, ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, ontap-san-economy	Tous
autoriser l'expansion du volume	booléen	vrai, faux	Activer ou désactiver la prise en charge de l'augmentation de la taille du PVC	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, ontap-san-economy, solidfire-san, gcp-cvs, azure-netapp-files	1,11+
mode de liaison du volume	chaîne	Immédiat, attendez le premier consommateur	Choisissez quand la liaison de volume et le provisionnement dynamique ont lieu.	Tous	1,19 - 1,26

- Le `fsType` Ce paramètre permet de contrôler le type de système de fichiers souhaité pour les LUN SAN. De plus, 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 de `fsGroup` contexte de sécurité d'un pod uniquement si `fsType` est réglé. Se référer à"Kubernetes : Configurer un contexte de sécurité pour un pod ou un conteneur" pour un aperçu de la configuration de la propriété des volumes à l'aide de `fsGroup` contexte. Kubernetes appliquera le `fsGroup` valeur uniquement si :

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

  Pour les pilotes de stockage NFS, un système de fichiers existe déjà dans le cadre de l'exportation NFS. Afin d'utiliser `fsGroup` la classe de stockage doit encore spécifier un `fsType` Vous pouvez le paramétrer pour `nfs` ou toute valeur non nulle.

- Se référer à"Augmenter les volumes" pour plus de détails sur l'expansion du volume.
- Le package d'installation de Trident fournit plusieurs exemples de définitions de classes de stockage à utiliser avec Trident `.sample-input/storage-class-*.yaml` . La suppression d'une classe de stockage Kubernetes entraîne également la suppression de la classe de stockage Trident correspondante.

Kubernetes `VolumeSnapshotClass` objets

Kubernetes `VolumeSnapshotClass` les objets sont analogues à `StorageClasses` . Ils permettent de définir plusieurs classes de stockage et sont référencés par les instantanés de volume pour associer l'instantané à la classe d'instantané requise. Chaque instantané de volume est associé à une seule classe d'instantané de volume.

UN `VolumeSnapshotClass` La création d'instantanés doit être définie par un administrateur. Une classe d'instantané 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` indique à Kubernetes que les demandes de snapshots de volume du `csi-snapclass` Les classes sont gérées par Trident. Le `deletionPolicy` Spécifie l'action à entreprendre lorsqu'un instantané doit être supprimé. Quand `deletionPolicy` est réglé sur `Delete` , les objets de snapshot de volume ainsi que le snapshot sous-jacent sur le cluster de stockage sont supprimés lorsqu'un snapshot est supprimé. Vous pouvez aussi le paramétrer sur `Retain` signifie que `VolumeSnapshotContent` et l'instantané physique sont conservés.

Kubernetes `VolumeSnapshot` objets

Un Kubernetes `VolumeSnapshot` L'objet est une requête visant à créer un instantané d'un volume. De même qu'un PVC représente une demande faite par un utilisateur pour un volume, un instantané de volume est une

demande faite par un utilisateur pour créer un instantané d'un PVC existant.

Lorsqu'une demande de snapshot de volume est reçue, Trident gère automatiquement la création du snapshot pour le volume sur le serveur et l'expose en créant un identifiant unique. `VolumeSnapshotContent` objet. Vous pouvez créer des instantanés à partir de PVC existants et utiliser ces instantanés comme source de données lors de la création de nouveaux PVC.

> Le cycle de vie d'un VolumeSnapshot est indépendant du PVC source : un snapshot persiste même après la suppression du PVC source. Lors de la suppression d'un PVC auquel sont associés des instantanés, Trident marque le volume de support de ce PVC dans un état **Suppression en cours**, mais ne le supprime pas complètement. Le volume est supprimé lorsque tous les instantanés associés sont supprimés.

Kubernetes `VolumeSnapshotContent` objets

Un Kubernetes `VolumeSnapshotContent` Cet objet représente un instantané pris à partir d'un volume déjà provisionné. Il est analogue à un `PersistentVolume` et désigne un instantané provisionné sur le cluster de stockage. Similaire à `PersistentVolumeClaim` et `PersistentVolume` objets, lors de la création d'un instantané, les `VolumeSnapshotContent` l'objet maintient une correspondance un-à-un avec le `VolumeSnapshot` objet, qui avait demandé la création de l'instantané.

Le `VolumeSnapshotContent` L'objet contient des détails qui identifient de manière unique la capture d'écran, tels que le `snapshotHandle` . Ce `snapshotHandle` est une combinaison unique du nom du PV et du nom du `VolumeSnapshotContent` objet.

Lorsqu'une demande de snapshot arrive, Trident crée le snapshot sur le serveur. Une fois l'instantané créé, Trident configure un `VolumeSnapshotContent` objet et expose ainsi l'instantané à l'API Kubernetes.

> En général, vous n'avez pas besoin de gérer le `VolumeSnapshotContent` objet. Une exception à cette règle est lorsque vous souhaitez"importer un instantané de volume" créé en dehors de Trident.

Kubernetes `VolumeGroupSnapshotClass` objets

Kubernetes `VolumeGroupSnapshotClass` les objets sont analogues à `VolumeSnapshotClass` . Ils permettent de définir plusieurs classes de stockage et sont référencés par les instantanés de groupes de volumes pour associer l'instantané à la classe d'instantané requise. Chaque instantané de groupe de volumes est associé à une seule classe d'instantané de groupe de volumes.

UN `VolumeGroupSnapshotClass` Un administrateur doit définir un groupe d'instantanés. Une classe d'instantané de groupe de volumes est créée avec la définition suivante :

```
apiVersion: groupsnapshot.storage.k8s.io/v1beta1
kind: VolumeGroupSnapshotClass
metadata:
  name: csi-group-snap-class
  annotations:
    kubernetes.io/description: "Trident group snapshot class"
driver: csi.trident.netapp.io
deletionPolicy: Delete
```

Le `driver` indique à Kubernetes que les demandes de snapshots de groupes de volumes du `csi-group-snap-class` Les classes sont gérées par Trident. Le `deletionPolicy` Spécifie l'action à entreprendre lorsqu'un instantané de groupe doit être supprimé. Quand `deletionPolicy` est réglé sur `Delete` , les objets d'instantané du groupe de volumes ainsi que l'instantané sous-jacent sur le cluster de stockage sont supprimés lorsqu'un instantané est supprimé. Vous pouvez aussi le paramétrer sur `Retain` signifie que `VolumeGroupSnapshotContent` et l'instantané physique sont conservés.

Kubernetes `VolumeGroupSnapshot` objets

Un Kubernetes `VolumeGroupSnapshot` L'objet est une requête visant à créer un instantané de plusieurs volumes. De même qu'un PVC représente une demande faite par un utilisateur pour un volume, un instantané de groupe de volumes est une demande faite par un utilisateur pour créer un instantané d'un PVC existant.

Lorsqu'une demande de snapshot de groupe de volumes est reçue, Trident gère automatiquement la création du snapshot de groupe pour les volumes sur le serveur et expose le snapshot en créant un identifiant unique. `VolumeGroupSnapshotContent` objet. Vous pouvez créer des instantanés à partir de PVC existants et utiliser ces instantanés comme source de données lors de la création de nouveaux PVC.

> Le cycle de vie d'un VolumeGroupSnapshot est indépendant du PVC source : un snapshot persiste même après la suppression du PVC source. Lors de la suppression d'un PVC auquel sont associés des instantanés, Trident marque le volume de support de ce PVC dans un état **Suppression en cours**, mais ne le supprime pas complètement. L'instantané du groupe de volumes est supprimé lorsque tous les instantanés associés sont supprimés.

Kubernetes `VolumeGroupSnapshotContent` objets

Un Kubernetes `VolumeGroupSnapshotContent` Cet objet représente un instantané de groupe pris à partir d'un volume déjà provisionné. Il est analogue à un `PersistentVolume` et désigne un instantané provisionné sur le cluster de stockage. Similaire à `PersistentVolumeClaim` et `PersistentVolume` objets, lors de la création d'un instantané, les `VolumeSnapshotContent` l'objet maintient une correspondance un-à-un avec le `VolumeSnapshot` objet, qui avait demandé la création de l'instantané.

Le `VolumeGroupSnapshotContent` L'objet contient des détails qui identifient le groupe d'instantanés, tels que le `volumeGroupSnapshotHandle` et les descripteurs Snapshot de volume individuels existants sur le système de stockage.

Lorsqu'une demande de snapshot arrive, Trident crée le snapshot du groupe de volumes sur le backend. Une fois l'instantané du groupe de volumes créé, Trident configure un `VolumeGroupSnapshotContent` objet et expose ainsi l'instantané à l'API Kubernetes.

Kubernetes `CustomResourceDefinition` objets

Les ressources personnalisées Kubernetes sont des points de terminaison 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 stocker une collection d'objets. Vous pouvez obtenir ces définitions de ressources en exécutant `kubectl get crds` .

Les définitions de ressources personnalisées (CRD) et leurs métadonnées d'objet associées sont stockées par Kubernetes dans son magasin de métadonnées. Cela élimine le besoin d'un magasin séparé pour Trident.

Trident utilise `CustomResourceDefinition` objets permettant de préserver l'identité des objets Trident , tels que les backends Trident , les classes de stockage Trident et les volumes Trident . Ces objets sont gérés par Trident. De plus, le cadre d'instantanés de volume CSI introduit certaines CRD nécessaires à la définition des instantanés de volume.

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

Voici quelques points à retenir concernant les CRD de Trident :

- Lors de l'installation de Trident , un ensemble de CRD est créé et peut être utilisé comme n'importe quel autre type de ressource.
- Lors de la désinstallation de Trident à l'aide de `tridentctl uninstall` La commande supprime les pods Trident , mais ne nettoie pas les CRD créés. Se référer à"Désinstaller Trident" pour comprendre comment Trident peut être entièrement supprimé et reconfiguré à partir de zéro.

Trident `StorageClass` objets

Trident crée des classes de stockage adaptées à Kubernetes `StorageClass` objets qui spécifient `csi.trident.netapp.io` dans leur domaine d'approvisionnement. Le nom de la classe de stockage correspond à celui de Kubernetes `StorageClass` objet qu'il représente.

> Avec Kubernetes, ces objets sont créés automatiquement lorsqu'un Kubernetes `StorageClass` qui utilise Trident comme fournisseur est enregistré.

Les classes de stockage comprennent un ensemble d'exigences relatives aux volumes. Trident compare ces exigences aux attributs présents dans chaque pool de stockage ; s'ils correspondent, ce pool de stockage est une cible valide pour le provisionnement de volumes utilisant cette classe de stockage.

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

Objets backend Trident

Les backends représentent les fournisseurs de stockage sur lesquels Trident provisionne les volumes ; une seule instance de Trident peut gérer un nombre quelconque de backends.

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

Pour plus d'informations sur la construction de ces objets, veuillez consulter"configuration des backends" .

Trident `StoragePool` objets

Les pools de stockage représentent les emplacements distincts disponibles pour le provisionnement sur chaque backend. Pour ONTAP, cela correspond à des agrégats dans les SVM. Pour NetApp HCI/ SolidFire, cela correspond à des bandes QoS spécifiées par l'administrateur. Pour Cloud Volumes Service, cela correspond aux régions des fournisseurs de cloud. Chaque pool de stockage possède un ensemble d'attributs de stockage distincts, qui définissent ses caractéristiques de performance et de protection des données.

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

Trident `Volume` objets

Les volumes constituent l'unité de base du provisionnement, comprenant des points de terminaison backend, tels que les partages NFS, et les LUN iSCSI et FC. Dans Kubernetes, ceux-ci correspondent directement à `PersistentVolumes` . Lorsque vous créez un volume, assurez-vous qu'il possède une classe de stockage, qui détermine où ce volume peut être provisionné, ainsi qu'une taille.

- Dans Kubernetes, ces objets sont gérés automatiquement. Vous pouvez les consulter pour voir ce que Trident a fourni.
- Lors de la suppression d'un PV avec des instantanés associés, le volume Trident correspondant est mis à jour à l'état **Suppression en cours**. Pour supprimer le volume Trident , vous devez supprimer les instantanés de ce volume.

Une configuration de volume définit les propriétés que doit posséder un volume provisionné.

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
cloneSourceVolume	chaîne	Non	ontap (nas, san) et solidfire-* : Nom du volume à cloner à partir de

Attribut	Type	Obligatoire	Description
splitOnClone	chaîne	Non	ontap (nas, san) : Séparer le clone de son parent
instantanéPolitique	chaîne	Non	ontap-* : Stratégie d'instantané à utiliser
snapshotReserve	chaîne	Non	ontap-* : Pourcentage du volume réservé aux instantanés
politique d'exportation	chaîne	Non	ontap-nas* : Politique d'exportation à utiliser
répertoire snapshot	booléen	Non	ontap-nas* : Indique si le répertoire des instantanés est visible
Autorisations Unix	chaîne	Non	ontap-nas* : Permissions UNIX initiales
taille du bloc	chaîne	Non	solidfire-*: Taille du bloc/secteur
système de fichiers	chaîne	Non	Type de système de fichiers

Trident génère `internalName` lors de la création du volume. Cela se compose de deux étapes. Premièrement, il ajoute le préfixe de stockage (soit le préfixe par défaut). `trident` ou le préfixe dans la configuration du backend) au nom du volume, ce qui donne un nom de la forme `<prefix>-<volume-name>` . Il procède ensuite à la désinfection du nom, en remplaçant les caractères non autorisés par le système. Pour les backends ONTAP , il remplace les tirets par des traits de soulignement (ainsi, le nom interne devient `<prefix>_<volume-name>`). Pour les backends Element, il remplace les underscores par des tirets.

Vous pouvez utiliser des configurations de volume pour provisionner directement des volumes via l'API REST, mais dans les déploiements Kubernetes, nous prévoyons que la plupart des utilisateurs utiliseront la configuration standard de Kubernetes. `PersistentVolumeClaim` méthode. Trident crée automatiquement cet objet volume dans le cadre du processus de provisionnement.

Trident `Snapshot` objets

Les snapshots sont une copie à un instant précis des volumes, qui peuvent être utilisés pour provisionner de nouveaux volumes ou restaurer un état. Dans Kubernetes, ceux-ci correspondent directement à `VolumeSnapshotContent` objets. Chaque instantané est associé à un volume, qui est la source des données de cet instantané.

Chaque `Snapshot` L'objet comprend les propriétés énumérées ci-dessous :

Attribut	Type	Obligatoire	Description
version	Chaîne	Oui	Version de l'API Trident (« 1 »)
nom	Chaîne	Oui	Nom de l'objet instantané Trident

Attribut	Type	Obligatoire	Description
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 l'instantané est créé
volumeInternalName	Chaîne	Oui	Nom de l'objet volume Trident associé sur le système de stockage

Dans Kubernetes, ces objets sont gérés automatiquement. Vous pouvez les consulter pour voir ce que Trident a fourni.

Lorsqu'un Kubernetes `VolumeSnapshot` Lorsqu'une requête d'objet est créée, Trident fonctionne en créant un objet instantané sur le système de stockage sous-jacent. Le `internalName` Cet objet instantané est généré en combinant le préfixe `snapshot-` avec le `UID` de la `VolumeSnapshot` objet (par exemple, `snapshot-e8d8a0ca-9826-11e9-9807-525400f3f660`). `volumeName` et `volumeInternalName` sont remplies en obtenant les détails du volume de support.

Trident `ResourceQuota` objet

Le démon Trident dévore un `system-node-critical` Classe de priorité – la classe de priorité la plus élevée disponible dans Kubernetes – pour garantir que Trident puisse identifier et nettoyer les volumes lors de l'arrêt progressif des nœuds et permettre aux pods du daemonset Trident de préempter les charges de travail de priorité inférieure dans les clusters où la pression sur les ressources est élevée.

Pour ce faire, Trident utilise un `ResourceQuota` objet pour garantir qu'une classe de priorité « critique pour le nœud système » sur le daemonset Trident est satisfaite. Avant le déploiement et la création du DaemonSet, Trident recherche le `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 le `ResourceQuota` objet utilisant le graphique Helm.

Voici un exemple d'objet `ResourceQuota` priorisant le daemonset 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, veuillez consulter"Kubernetes : Quotas de ressources" .

Nettoyage `ResourceQuota` si l'installation échoue

Dans les rares cas où l'installation échoue après le `ResourceQuota` L'objet est créé, première tentative"désinstallation" puis réinstaller.

Si cela ne fonctionne pas, retirez manuellement le `ResourceQuota` objet.

Retirer `ResourceQuota`

Si vous préférez gérer vous-même l'allocation de vos ressources, vous pouvez retirer le Trident. `ResourceQuota` objet utilisant la commande :

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

Normes de sécurité des modules (PSS) et contraintes de contexte de sécurité (SCC)

Les normes de sécurité des pods Kubernetes (PSS) et les politiques de sécurité des pods (PSP) définissent les niveaux d'autorisation et restreignent le comportement des pods. Les contraintes de contexte de sécurité OpenShift (SCC) définissent de la même manière les restrictions de pod spécifiques au moteur Kubernetes d'OpenShift. Pour permettre cette personnalisation, Trident active certaines autorisations lors de l'installation. Les sections suivantes détaillent les autorisations définies par Trident.

PSS remplace les politiques de sécurité des pods (PSP). PSP a été déprécié dans Kubernetes v1.21 et sera supprimé dans la v1.25. Pour plus d'informations, veuillez consulter"Kubernetes : Sécurité" .

Contexte de sécurité Kubernetes requis et champs associés

Autorisation	Description
Privilégié	CSI exige que les points de montage soient bidirectionnels, ce qui signifie que le pod du nœud Trident doit exécuter un conteneur privilégié. Pour plus d'informations, veuillez consulter"Kubernetes : Propagation du montage" .
Réseau hôte	Requis pour le démon iSCSI. `iscsiadm` Gère les montages iSCSI et utilise le réseau hôte pour communiquer avec le démon iSCSI.
IPC de l'hôte	NFS utilise la communication interprocessus (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` s'exécute avant le montage des volumes NFS.
Capacités	Le `SYS_ADMIN` Cette fonctionnalité est fournie dans le cadre des fonctionnalités par défaut des conteneurs privilégiés. Par exemple, Docker définit les capacités suivantes pour les conteneurs privilégiés : `CapPrm: 0000003fffffffff` `CapEff: 0000003fffffffff`
Seccomp	Le profil Seccomp est toujours « non confiné » dans les conteneurs privilégiés ; par conséquent, il ne peut pas être activé dans Trident.
SELinux	Sur OpenShift, les conteneurs privilégiés sont exécutés dans le `spc_t` Le domaine (« Super Privileged Container ») et les conteneurs non privilégiés sont exécutés dans le domaine `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 en tant que root pour accéder aux sockets Unix requis par CSI.

Normes de sécurité des capsules (PSS)

Étiquette	Description	Défaut
`pod-security.kubernetes.io/enforce pod-security.kubernetes.io/enforce-version`	Permet d'admettre le contrôleur Trident et les nœuds dans l'espace de noms d'installation. Ne modifiez pas l'étiquette de l'espace de noms.	`enforce: privileged enforce-version: <version of the current cluster or highest version of PSS tested.>`

> Modifier les étiquettes d'espace de noms peut empêcher la planification des pods, entraîner une erreur lors de la création de : … ou un avertissement : trident-csi-…. Si cela se produit, vérifiez si l'étiquette d'espace de noms pour `privileged` a été modifié. Si c'est le cas, réinstallez Trident.

Politiques de sécurité des pods (PSP)

Champ	Description	Défaut
`allowPrivilegeEscalation`	Les conteneurs privilégiés doivent autoriser l'élévation de privilèges.	`true`
`allowedCSIDrivers`	Trident n'utilise pas de volumes éphémères CSI en ligne.	Vide
`allowedCapabilities`	Les conteneurs Trident non privilégiés ne nécessitent pas plus de capacités que l'ensemble par défaut, tandis que les conteneurs privilégiés se voient accorder toutes les capacités possibles.	Vide
`allowedFlexVolumes`	Trident n'utilise pas de"Pilote FlexVolume" , par conséquent, ils ne sont pas inclus dans la liste des volumes autorisés.	Vide
`allowedHostPaths`	Le pod du nœud Trident monte le système de fichiers racine du nœud ; par conséquent, la configuration de cette liste ne présente aucun avantage.	Vide
`allowedProcMountTypes`	Trident n'utilise aucun `ProcMountTypes` .	Vide
`allowedUnsafeSysctls`	Trident ne nécessite aucune protection contre les risques. `sysctls` .	Vide
`defaultAddCapabilities`	Aucune fonctionnalité supplémentaire n'est requise pour les conteneurs privilégiés.	Vide
`defaultAllowPrivilegeEscalation`	L'autorisation d'élévation de privilèges est gérée dans chaque pod Trident .	`false`
`forbiddenSysctls`	Non `sysctls` sont autorisés.	Vide

Champ	Description	Défaut
`fsGroup`	Les conteneurs Trident s'exécutent en tant que root.	`RunAsAny`
`hostIPC`	Le montage de volumes NFS nécessite que l'IPC de l'hôte communique avec `nfsd`	`true`
`hostNetwork`	iscsiadm nécessite que le réseau hôte communique avec le démon iSCSI.	`true`
`hostPID`	Le PID de l'hôte est requis pour vérifier si `rpc-statd` est en cours d'exécution sur le nœud.	`true`
`hostPorts`	Trident n'utilise aucun port hôte.	Vide
`privileged`	Les pods de nœuds Trident doivent exécuter un conteneur privilégié pour pouvoir monter des volumes.	`true`
`readOnlyRootFilesystem`	Les pods de nœud Trident doivent écrire sur le système de fichiers du nœud.	`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`
`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 manière dont les environnements d'exécution de conteneurs et les distributions Kubernetes gèrent SELinux.	Vide
`supplementalGroups`	Les conteneurs Trident s'exécutent en tant que root.	`RunAsAny`
`volumes`	Les modules Trident nécessitent ces plugins de volume.	`hostPath, projected, emptyDir`

Contraintes de contexte de sécurité (CCS)

Étiquettes	Description	Défaut
`allowHostDirVolumePlugin`	Les pods de nœud Trident montent le système de fichiers racine du nœud.	`true`
`allowHostIPC`	Le montage de volumes NFS nécessite que l'IPC de l'hôte communique avec `nfsd` .	`true`
`allowHostNetwork`	iscsiadm nécessite que le réseau hôte communique avec le démon iSCSI.	`true`
`allowHostPID`	Le PID de l'hôte est requis pour vérifier si `rpc-statd` est en cours d'exécution sur le nœud.	`true`
`allowHostPorts`	Trident n'utilise aucun port hôte.	`false`
`allowPrivilegeEscalation`	Les conteneurs privilégiés doivent autoriser l'élévation de privilèges.	`true`
`allowPrivilegedContainer`	Les pods de nœuds Trident doivent exécuter un conteneur privilégié pour pouvoir monter des volumes.	`true`
`allowedUnsafeSysctls`	Trident ne nécessite aucune protection contre les risques. `sysctls` .	`none`
`allowedCapabilities`	Les conteneurs Trident non privilégiés ne nécessitent pas plus de capacités que l'ensemble par défaut, tandis que les conteneurs privilégiés se voient accorder toutes les capacités possibles.	Vide
`defaultAddCapabilities`	Aucune fonctionnalité supplémentaire n'est requise pour les conteneurs privilégiés.	Vide
`fsGroup`	Les conteneurs Trident s'exécutent en tant que root.	`RunAsAny`
`groups`	Ce SCC est spécifique à Trident et est lié à son utilisateur.	Vide
`readOnlyRootFilesystem`	Les pods de nœud Trident doivent écrire sur le système de fichiers du nœud.	`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`

Étiquettes	Description	Défaut
`seLinuxContext`	Trident ne définit pas `seLinuxOptions` car il existe actuellement des différences dans la manière dont les environnements d'exécution de conteneurs et les distributions Kubernetes gèrent SELinux.	Vide
`seccompProfiles`	Les conteneurs privilégiés s'exécutent toujours en mode « non confiné ».	Vide
`supplementalGroups`	Les conteneurs Trident s'exécutent en tant que root.	`RunAsAny`
`users`	Une entrée est prévue pour lier ce SCC à l'utilisateur Trident dans l'espace de noms Trident .	n / A
`volumes`	Les modules Trident nécessitent ces plugins de volume.	`hostPath, downwardAPI, projected, emptyDir`