Référence

Astra Trident

NetApp
April 04, 2024

Sommaire

Référence

Ports Trident d'Astra

Découvrez les ports qu'Astra Trident utilise pour la communication.

Ports Trident d'Astra

Astra 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 de liaison/préparation peut être modifié lors de l'installation à l'aide du `--probe-port` drapeau. Il est important de s'assurer que ce port n'est pas utilisé par un autre processus sur les nœuds worker.

API REST d'Astra Trident

Pendant "commandes et options tridentctl" Vous pouvez utiliser le terminal REST directement si vous le souhaitez. Pour interagir avec l'API REST Astra Trident,

Quand utiliser l'API REST

Il est utile pour les installations avancées qui utilisent Astra Trident en tant que binaire autonome dans les déploiements non Kubernetes.

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

Avec l'API REST

L'API fonctionne comme suit :

`GET`

- `GET <trident-address>/trident/v1/<object-type>`: Affiche 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, voir lien:tridentctl.html[`tridentctl` commandes et options].
- 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.

`DELETE`

`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, consultez le lien :tridentctl.html[`tridentctl` commandes et options].

Pour des exemples de la façon dont ces API sont appelées, passez le débogage (`-d`) drapeau. Pour plus d'informations, consultez le lien :tridentctl.html[`tridentctl` commandes et options].

Options de ligne de commande

Astra 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 consignation (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. Trident peut donc être déployé en dehors d'un pod, mais il ne prend uniquement en charge les connexions non sécurisées vers le serveur API. Pour vous connecter de manière sécurisée, déployez Trident dans un pod avec le `-k8s_pod` option.
- `-k8s_config_path <file>`: Obligatoire ; vous devez spécifier ce chemin d'accès à un fichier KubeConfig.

Docker

- `-volume_driver <name>`: Nom du pilote utilisé lors de l'enregistrement du plugin Docker. La valeur par

défaut est `netapp`.

- `-driver_port <port-number>`: Écouter 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 à laquelle le serveur DE REPOS 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. Utiliser `-address ""` Pour rendre l'interface REST accessible depuis 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>`: Indique 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.

Produits NetApp intégrés avec Kubernetes

Le portefeuille de produits de stockage NetApp s'intègre à différents aspects d'un cluster Kubernetes, offrant des fonctionnalités avancées de gestion des données qui améliorent la fonctionnalité, la capacité, la performance et la disponibilité du déploiement Kubernetes.

Astra

"Astra" Les entreprises peuvent ainsi gérer, protéger et déplacer plus facilement leurs workloads conteneurisés dans des clouds publics et sur site, que ce soit sur Kubernetes. Astra provisionne et fournit un stockage persistant pour les conteneurs en utilisant Trident, la gamme de stockage étendue et éprouvée de NetApp dans le cloud public et sur site. Il offre également un ensemble complet de fonctionnalités avancées de gestion des données intégrant la cohérence applicative, telles que les copies Snapshot, la sauvegarde et la restauration, les journaux d'activité et le clonage actif pour la protection des données, la reprise d'activité, l'audit des données et la migration pour les workloads Kubernetes.

ONTAP

ONTAP est un système d'exploitation de stockage unifié multiprotocole de NetApp qui offre des fonctionnalités avancées de gestion des données pour toutes les applications. Les systèmes ONTAP sont dotés de configurations 100 % Flash, hybrides ou 100 % HDD et proposent différents modèles de déploiement, notamment du matériel spécialisé (FAS et AFF), de l'infrastructure générique (ONTAP Select) et du cloud uniquement (Cloud Volumes ONTAP).

> Trident prend en charge tous les modèles de déploiement ONTAP mentionnés ci-dessus.

Cloud Volumes ONTAP

"Cloud Volumes ONTAP" Est une appliance de stockage exclusivement logicielle qui exécute le logiciel de gestion des données ONTAP dans le cloud. Vous pouvez utiliser Cloud Volumes ONTAP pour les charges de travail de production, la reprise après incident, les DevOps, les partages de fichiers et la gestion des bases de données. Il étend le stockage d'entreprise au cloud en offrant les fonctionnalités d'efficacité du stockage, la haute disponibilité, la réplication des données, le Tiering des données et la cohérence applicative.

Amazon FSX pour NetApp ONTAP

"Amazon FSX pour NetApp ONTAP" 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 NetApp ONTAP. La solution FSX pour ONTAP permet aux clients d'exploiter les fonctionnalités, les performances et les capacités d'administration de NetApp qu'ils connaissent, tout en profitant de la simplicité, de l'agilité, de la sécurité et de l'évolutivité du stockage des données sur AWS. FSX pour ONTAP prend en charge de nombreuses fonctionnalités de système de fichiers et API d'administration d'ONTAP.

Logiciel Element

"Elément" offre à l'administrateur du stockage la possibilité de consolider les charges de travail pour un encombrement du stockage simplifié et optimisé. Associé à une API pour automatiser tous les aspects de la gestion du stockage, Element permet aux administrateurs du stockage d'en faire plus avec moins d'efforts.

NetApp HCI

"NetApp HCI" simplifie la gestion et l'évolutivité du data center en automatisant les tâches de routine et en permettant aux administrateurs d'infrastructure de se concentrer sur des fonctions plus importantes.

NetApp HCI est entièrement pris en charge par Trident. Trident peut provisionner et gérer des terminaux de stockage pour les applications conteneurisées directement sur la plateforme de stockage NetApp HCI sous-jacente.

Azure NetApp Files

"Azure NetApp Files" Est un service de partage de fichiers Azure haute performance optimisé par NetApp. Vous pouvez exécuter les workloads basés sur des fichiers les plus exigeants dans Azure de façon native, avec les performances et les fonctionnalités avancées de gestion des données que vous attendez de NetApp.

Cloud Volumes Service pour Google Cloud

"NetApp Cloud Volumes Service pour Google Cloud" Est un service de fichiers cloud natif qui fournit des volumes NAS sur NFS et SMB avec des performances 100 % Flash. Ce service permet à tous les workloads, y compris les applications héritées, d'être exécutés dans le cloud GCP. Il fournit un service entièrement géré qui offre des performances élevées et régulières, un clonage instantané, une protection des données et un accès sécurisé aux instances Google Compute Engine (GCE).

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 un `PersistentVolumeClaim` demander un nouveau `PersistentVolume` D'une taille spécifique dans un Kubernetes `StorageClass` qui a été précédemment configuré par l'administrateur.
2. Le Kubernetes `StorageClass` Identifie Trident comme mécanisme de provisionnement et inclut des paramètres indiquant à Trident comment provisionner un volume pour la classe demandée.
3. Trident s'occupe par lui-même `StorageClass` avec 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 back-end correspondant et crée deux objets : un `PersistentVolume` Dans Kubernetes qui indique à Kubernetes comment rechercher, monter et traiter le volume, et à un volume dans Trident qui conserve la relation entre le système `PersistentVolume` et le stockage réel.
5. Kubernetes lie le `PersistentVolumeClaim` vers le nouveau `PersistentVolume`. Des modules qui incluent `PersistentVolumeClaim` Montez le volume persistant sur n'importe quel hôte sur lequel il s'exécute.
6. Un utilisateur crée un `VolumeSnapshot` D'un volume persistant existant, à l'aide d'un `VolumeSnapshotClass` Ce que nous 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` Cela indique à Kubernetes comment identifier le Snapshot.
8. Un utilisateur peut créer un `PersistentVolumeClaim` à l'aide de `VolumeSnapshot` en tant que source.
9. Trident identifie le snapshot requis et effectue les mêmes étapes que lors de la création d'un `PersistentVolume` et a `Volume`.

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

Kubernetes `PersistentVolumeClaim` objets

Un Kubernetes `PersistentVolumeClaim` Cet objet est une demande de stockage faite par un utilisateur du 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

Annotation	Option de volume	Pilotes pris en charge
trident.netapp.io/cloneFromPVC	Volume cloneSourceVolume	nas ontap, 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	nas ontap, ontap-nas-economy, ontap-nas-flexgroup
trident.netapp.io/snapshotPolicy	Politique de snapshots	nas ontap, économie ontap-nas, ontap-nas-flexgroup, ontap-san
trident.netapp.io/snapshotReserve	Réserve de snapshots	nas ontap, ontap-nas-flexgroup, ontap-san, gcp-cvs
trident.netapp.io/snapshotDirectory	Répertoire de snapshots	nas ontap, ontap-nas-economy, ontap-nas-flexgroup
trident.netapp.io/unixPermissions	Autorisations unix	nas ontap, ontap-nas-economy, ontap-nas-flexgroup
trident.netapp.io/blockSize	Taille de bloc	solidfire-san

Si le volume persistant créé est de `Delete` Lors de la récupération de la règle, Trident supprime le volume persistant et le volume de sauvegarde lorsque le volume persistant est libéré (c'est-à-dire lors de la suppression de 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 PV utilise le `Retain` La règle, Trident l'ignore et suppose que l'administrateur l'nettoie depuis Kubernetes et le back-end, permettant ainsi 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. Découvrez-en plus `On-Demand Volume Snapshots` pour voir comment cela fonctionne.

Trident fournit également le système `cloneFromPVC` et `splitOnClone` annotations pour la création de clones. Vous pouvez utiliser ces annotations pour cloner une demande de volume persistant sans avoir à utiliser l'implémentation CSI (sur Kubernetes 1.13 et versions antérieures) ou si votre version Kubernetes ne prend pas en charge les copies Snapshot de volume bêta (Kubernetes 1.16 et versions antérieures). N'oubliez pas que Trident 19.10 prend en charge le workflow CSI pour le clonage à partir d'une demande de volume persistant.

Vous pouvez utiliser le `cloneFromPVC` et `splitOnClone` Annotations avec CSI Trident ainsi que le front-end traditionnel non CSI.

Voici un exemple : si un utilisateur a déjà un volume persistant appelé `mysql`, L'utilisateur peut créer un nouveau PVC appelé `mysqlclone` en utilisant l'annotation, par exemple `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 :

- Nous vous recommandons 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 le `ontap-nas` et `ontap-san` Pilotes, il peut être souhaitable de définir l'annotation PVC `trident.netapp.io/splitOnClone` en conjonction avec `trident.netapp.io/cloneFromPVC`. Avec `trident.netapp.io/splitOnClone` réglez sur `true`, Trident divise le volume cloné du volume parent et, par conséquent, découplant complètement le cycle de vie du volume cloné de sa parent, au détriment de la perte de l'efficacité du stockage. Pas de réglage `trident.netapp.io/splitOnClone` ou le définir sur `false` cette baisse de la consommation d'espace sur le back-end implique des frais de création des dépendances entre les volumes parent et clone de sorte que le volume parent ne puisse pas être supprimé, à moins que le clone ne soit 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` Le répertoire contient des exemples de définitions de volume persistant à utiliser avec Trident. Pour une description complète des paramètres et des paramètres associés aux volumes Trident, reportez-vous à la section objets volume Trident.

Kubernetes `PersistentVolume` objets

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

Création de Trident `PersistentVolume` Les objets et les enregistre automatiquement avec 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 persistant faisant référence à une configuration 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.

Kubernetes `StorageClass` objets

Kubernetes `StorageClass` les objets sont spécifiés par le nom dans `PersistentVolumeClaims` 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.

Un Kubernetes `StorageClass` Voici quelques aspects d'un objet qui utilise 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	Mapper les noms de back-end aux listes de pools de stockage dans
Des médutiquesde stockage	Mapper[string]StringList	non	Carte des noms de back-end pour afficher les listes de pools de stockage dans
Exclus du stockagePools	Mapper[string]StringList	non	Mapper les noms de back-end à listes des 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
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
D'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 l'utiliser `attributes` modélisez les qualités de stockage dont vous avez besoin pour répondre à vos besoins. Trident détecte et sélectionne automatiquement les pools de stockage qui correspondent à *All* du `attributes` que vous spécifiez.

Si vous vous trouvez incapable d'utiliser `attributes` pour sélectionner automatiquement les pools appropriés pour une classe, vous pouvez utiliser le `storagePools` et `additionalStoragePools` paramètres 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 spécifié `attributes`. En d'autres termes, Trident utilise l'intersection des pools identifiés par le `attributes` et `storagePools` paramètres de provisionnement. Vous pouvez utiliser les paramètres seuls ou les deux ensemble.

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

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

Dans le `storagePools` et `additionalStoragePools` paramètres, 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-être cela `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	Kubernetes Version
Fstype	chaîne	ext4, ext3, xfs, etc	Type de système de fichiers pour le bloc volumes	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. Kubernetes utilise également la présence de `fsType` dans une classe de stockage pour indiquer qu'un système de fichiers existe. Vous pouvez contrôler la propriété de volume à l'aide du `fsGroup` contexte de sécurité d'un pod uniquement si `fsType` est défini. Voir "Kubernetes : configurez un contexte de sécurité pour un pod ou un conteneur" pour une vue d'ensemble de la définition de la propriété de volume à l'aide de l' `fsGroup` contexte. Kubernetes applique le `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 l'utilisation `fsGroup` la classe de stockage doit toujours spécifier un `fsType`. Vous pouvez le définir sur `nfs` ou toute valeur non nulle.
- Voir "Développement des volumes" pour plus de détails sur l'extension du volume.
- Le bundle d'installation Trident propose plusieurs exemples de définitions de classes 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.

Kubernetes `VolumeSnapshotClass` objets

Kubernetes `VolumeSnapshotClass` les objets sont similaires 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.

A `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 demande des snapshots de volume du `csi-snapclass` Ces classes sont gérées par Trident. Le `deletionPolicy` spécifie l'action à effectuer lorsqu'un instantané doit être supprimé. Quand `deletionPolicy` est défini sur `Delete`, les objets de snapshot de volume ainsi que le snapshot sous-jacent du cluster de stockage sont supprimés lorsqu'un snapshot est supprimé. Vous pouvez également le régler sur `Retain` signifie que `VolumeSnapshotContent` et le snapshot physique sont conservés.

Kubernetes `VolumeSnapshot` objets

Un Kubernetes `VolumeSnapshot` objet est une demande de création d'un snapshot de 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 requête de snapshot de volume est fournie, Trident gère automatiquement la création du snapshot du volume sur le back-end et expose le snapshot en créant un seul snapshot `VolumeSnapshotContent` objet. 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.

Kubernetes `VolumeSnapshotContent` objets

Un Kubernetes `VolumeSnapshotContent` objet représente un snapshot pris à partir d'un volume déjà provisionné. Il est similaire à un `PersistentVolume` la désignation rr signifie un snapshot provisionné sur le cluster de stockage. Similaire à `PersistentVolumeClaim` et `PersistentVolume` lors de la création d'un snapshot, le `VolumeSnapshotContent` l'objet conserve un mappage un-à-un avec le `VolumeSnapshot` objet, qui avait demandé la création de snapshot.

> Création de Trident `VolumeSnapshotContent` Les objets et les enregistre automatiquement avec le cluster Kubernetes en fonction des volumes qu'il provisionne. Vous n'êtes pas censé les gérer vous-même.

Le `VolumeSnapshotContent` l'objet contient des détails qui identifient de manière unique le snapshot, comme le `snapshotHandle`. C'est ça `snapshotHandle` Est une combinaison unique du nom du PV et du nom du `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 donc expose le snapshot à l'API Kubernetes.

Kubernetes `CustomResourceDefinition` objets

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 cours d'exécution `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 également la version 19.07 de `CustomResourceDefinition` 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. À titre d'exemple simple, lorsqu'un système back-end est créé à l'aide de `tridentctl`, un correspondant `tridentbackends` L'objet CRD est créé pour la consommation 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 mise à niveau à partir d'une version précédente de Trident (celle qui était utilisée) `etcd` Pour préserver l'état), le programme d'installation de Trident migre les données du système `etcd` Le stockage de données à clé-valeur et crée les objets CRD correspondants.
- Lors de la désinstallation de Trident à l'aide de `tridentctl uninstall` Les pods Trident sont supprimés, mais les CRD créés ne sont pas nettoyés. Voir "Désinstaller Trident" Afin de comprendre comment Trident peut être entièrement supprimé et reconfiguré de zéro.

Trident `StorageClass` objets

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

> Avec Kubernetes, ces objets sont créés automatiquement lorsqu'un système Kubernetes est activé `StorageClass` Qui utilise Trident comme mécanisme de provisionnement 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, dans le cas des déploiements Kubernetes, nous attendons d'eux qu'ils soient créés lors de l'enregistrement du nouveau Kubernetes `StorageClass` objets.

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 le Kubernetes `StorageClass` objet.

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

Trident `StoragePool` objets

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.

Trident `Volume` objets

Les volumes sont l'unité de provisionnement de base, comprenant les terminaux back-end, tels que les partages NFS et les LUN iSCSI. Dans Kubernetes, ces derniers 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

Attribut	Type	Obligatoire	Description
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
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

Génération de Trident `internalName` lors de la création du volume. Il s'agit de deux étapes. Tout d'abord, il prétermine le préfixe de stockage (soit le préfixe par défaut `trident` ou le préfixe de la configuration back-end) au nom du volume, ce qui produit un nom du 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 (ainsi, le nom interne devient `<prefix>_<volume-name>`). Pour les systèmes back-end Element, il remplace les tirets de traits de soulignement.

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

Trident `Snapshot` objets

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 derniers correspondent directement à `VolumeSnapshotContent` objets. Chaque snapshot est associé à un volume, qui est la source des données du snapshot.

Chacun `Snapshot` l'objet inclut 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.

Lorsqu'un Kubernetes `VolumeSnapshot` La requête d'objet est créée, Trident crée un objet de snapshot sur le système de stockage secondaire. Le `internalName` cet objet de snapshot est généré en combinant le préfixe `snapshot-` avec le `UID` du `VolumeSnapshot` objet (par exemple, `snapshot-e8d8a0ca-9826-11e9-9807-525400f3f660`). `volumeName` et `volumeInternalName` sont renseignés en obtenant les détails du support
volumétrie.

Astra Trident `ResourceQuota` objet

La déamOnset Trident utilise un `system-node-critical` Classe de priorité—la classe de priorité la plus élevée disponible dans Kubernetes—pour s'assurer que Astra Trident peut identifier et nettoyer les volumes lors de l'arrêt normal des nœuds. Il permet également aux pods Trident de s'assurer que ces derniers anticipent les charges de travail dans les clusters où la pression des ressources est élevée.

Astra Trident utilise un pour atteindre ces objectifs `ResourceQuota` Objet garantissant la satisfaction d'une classe de priorité « système-nœud-critique » sur le jeu de démonéset Trident. Avant de déployer et de diaboset, Astra 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 configurez le `ResourceQuota` Objet utilisant le 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 à la section "Kubernetes : quotas de ressources".

Nettoyez `ResourceQuota` si l'installation échoue

Dans les rares cas où l'installation échoue après le `ResourceQuota` l'objet est créé, commencez par essayer "désinstallation" puis réinstaller.

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

Déposer `ResourceQuota`

Si vous préférez contrôler l'allocation de vos ressources, vous pouvez supprimer Astra Trident `ResourceQuota` objet utilisant la commande :

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

commandes et options tridentctl

Le "Pack d'installation Trident" inclut un utilitaire de ligne de commande, `tridentctl`, Il fournit un accès simple à Astra Trident. Les utilisateurs de Kubernetes avec suffisamment de privilèges peuvent l'utiliser pour installer Astra Trident et interagir directement avec lui afin de gérer le namespace contenant le pod Astra Trident.

Commandes et options disponibles

Pour les informations d'utilisation, exécutez `tridentctl --help`.

Les commandes disponibles et les options globales sont les suivantes :

```
Usage:
  tridentctl [command]
```

Commandes disponibles :

- `create`: Ajouter une ressource à Astra Trident.
- `delete`: Supprimer une ou plusieurs ressources d'Astra Trident.
- `get`: Obtenez une ou plusieurs ressources d'Astra Trident.
- `help`: Aide sur toute commande.
- `images`: Imprimer un tableau des images de conteneur dont Astra Trident a besoin.
- `import`: Importer une ressource existante vers Astra Trident.
- `install`: Installer Astra Trident.
- `logs`: Imprimer les journaux d'Astra Trident.
- `send`: Envoyer une ressource d'Astra Trident.
- `uninstall`: Désinstaller Astra Trident.
- `update`: Modifier une ressource dans Astra Trident.
- `upgrade`: Mettre à niveau une ressource dans Astra Trident.
- `version`: Imprimer la version d'Astra Trident.

Alarmes :

- `-d, --debug`: Sortie de débogage.
- `-h, --help`: Aide pour `tridentctl`.
- `-n, --namespace string`: Espace de noms du déploiement Astra Trident.
- `-o, --output string`: Format de sortie. Un de json|yaml|nom|large|ps (par défaut).
- `-s, --server string`: Adresse/port de l'interface REST Astra Trident.

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.

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.

`create`

Vous pouvez utiliser le `create` Commande d'ajout d'une ressource à Astra Trident.

```
Usage:
  tridentctl create [option]
```

Option disponible :
`backend`: Ajouter un backend à Astra Trident.

delete

Vous pouvez exécuter le `delete` Commande de supprimer une ou plusieurs ressources d'Astra Trident.

```
Usage:
  tridentctl delete [option]
```

Options disponibles :

- `backend`: Supprimer un ou plusieurs systèmes back-end de stockage d'Astra Trident.
- `snapshot`: Supprimez un ou plusieurs instantanés de volume d'Astra Trident.
- `storageclass`: Supprimez une ou plusieurs classes de stockage d'Astra Trident.
- `volume`: Supprimer un ou plusieurs volumes de stockage d'Astra Trident.

get

Vous pouvez exécuter le `get` Commander pour obtenir une ou plusieurs ressources d'Astra Trident.

```
Usage:
  tridentctl get [option]
```

Options disponibles :

- `backend`: Faites passer un ou plusieurs systèmes back-end de stockage à Astra Trident.
- `snapshot`: Obtenez un ou plusieurs instantanés d'Astra Trident.
- `storageclass`: Obtenez une ou plusieurs classes de stockage d'Astra Trident.
- `volume`: Obtenez un ou plusieurs volumes d'Astra Trident.

`volume` indicateurs :
* `-h, --help`: Aide pour les volumes.
* `--parentOfSubordinate string`: Limiter la requête au volume source subordonné.
* `--subordinateOf string`: Limiter la requête aux subordonnés du volume.

images

Vous pouvez exécuter le `images` Drapeau pour imprimer un tableau des images de conteneur dont Astra

Trident a besoin.

```
Usage:
  tridentctl images [flags]
```

Alarmes :
* `-h, --help`: Help for images.
* -V, --k8s-version string`: Version sémantique du cluster Kubernetes.

import volume

Vous pouvez exécuter le `import volume` Commande permettant d'importer un volume existant vers Astra Trident.

```
Usage:
  tridentctl import volume <backendName> <volumeName> [flags]
```

Alias :
`volume, v`

Alarmes :

- `-f, --filename string`: Chemin vers le fichier PVC YAML ou JSON.
- `-h, --help`: Aide pour le volume.
- `--no-manage`: Créer PV/PVC uniquement. Ne supposez pas la gestion du cycle de vie des volumes.

install

Vous pouvez exécuter le `install` Drapeaux pour l'installation d'Astra Trident.

```
Usage:
  tridentctl install [flags]
```

Alarmes :

- `--autosupport-image string`: L'image du conteneur pour la télémétrie AutoSupport (par défaut « NetApp/trident autosupport :20.07.0 »).
- `--autosupport-proxy string`: Adresse/port d'un proxy pour l'envoi de télémétrie AutoSupport.
- `--csi`: Installer CSI Trident (remplacer pour Kubernetes 1.13 uniquement, nécessite des grilles de fonction).
- `--enable-node-prep`: Tentative d'installation des paquets requis sur les nœuds.
- `--generate-custom-yaml`: Générer des fichiers YAML sans rien installer.
- `-h, --help`: Aide pour l'installation.

- `--http-request-timeout`: Remplacer le délai de requête HTTP pour l'API REST du contrôleur Trident (par défaut 1m30s).
- `--image-registry string`: Adresse/port d'un registre d'images interne.
- `--k8s-timeout duration`: Délai d'expiration pour toutes les opérations Kubernetes (par défaut 3m0s).
- `--kubelet-dir string`: L'emplacement hôte de l'état interne de kubelet (par défaut "/var/lib/kubelet").
- `--log-format string`: Le format de consignation Astra Trident (texte, json) (par défaut "texte").
- `--pv string`: Le nom de la PV héritée utilisée par Astra Trident, s'assure que cela n'existe pas (par défaut "trident").
- `--pvc string`: Le nom du PVC hérité utilisé par Astra Trident, s'assure qu'il n'existe pas (par défaut "trident").
- `--silence-autosupport`: N'envoyez pas automatiquement les packs AutoSupport à NetApp (valeur par défaut vraie).
- `--silent`: Désactiver la plupart des sorties lors de l'installation.
- `--trident-image string`: L'image Astra Trident à installer.
- `--use-custom-yaml`: Utilisez tous les fichiers YAML existants qui existent dans le répertoire de configuration.
- `--use-ipv6`: Utiliser IPv6 pour la communication d'Astra Trident.

logs

Vous pouvez exécuter le `logs` Drapeaux pour imprimer les journaux à partir d'Astra Trident.

```
Usage:
  tridentctl logs [flags]
```

Alarmes :

- `-a, --archive`: Créez une archive de support avec tous les journaux sauf indication contraire.
- `-h, --help`: Aide pour les journaux.
- `-l, --log string`: Astra Trident log à afficher. Un de trident|auto|trident-operator|All (auto par défaut).
- `--node string`: Le nom du nœud Kubernetes à partir duquel recueillir les journaux de pod de nœud.
- `-p, --previous`: Si elle existe, procurez-vous les journaux de l'instance de conteneur précédente.
- `--sidecars`: Procurez-vous les bûches pour les conteneurs de sidecar.

send

Vous pouvez exécuter le `send` Commande permettant d'envoyer une ressource à Astra Trident.

```
Usage:
  tridentctl send [option]
```

Option disponible :
`autosupport`: Envoyez une archive AutoSupport à NetApp.

uninstall

Vous pouvez exécuter le `uninstall` Drapeaux pour désinstaller Astra Trident.

```
Usage:
  tridentctl uninstall [flags]
```

Alarmes :
* `-h, --help`: Aide pour désinstaller.
* `--silent`: Désactiver la plupart des résultats lors de la désinstallation.

update

Vous pouvez exécuter le `update` Commandes permettant de modifier une ressource dans Astra Trident.

```
Usage:
  tridentctl update [option]
```

Options disponibles :
`backend`: Mettre à jour un backend dans Astra Trident.

upgrade

Vous pouvez exécuter le `upgrade` Commandes de mise à niveau d'une ressource dans Astra Trident.

```
Usage:
tridentctl upgrade [option]
```

Option disponible :
`volume`: Mettre à niveau un ou plusieurs volumes persistants de NFS/iSCSI vers CSI.

version

Vous pouvez exécuter le `version` indicateurs pour imprimer la version de `tridentctl` Et le service exécutant Trident.

```
Usage:
  tridentctl version [flags]
```

Alarmes :
* `--client`: Version client uniquement (aucun serveur requis).
* `-h, --help`: Aide pour la version.

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 offrir cette personnalisation, Astra Trident autorise certaines autorisations lors de l'installation. Les sections suivantes décrivent en détail les autorisations définies par Astra 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, voir "Kubernetes : sécurité".

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, voir "Kubernetes : propagation du montage".
Mise en réseau d'hôtes	Requis pour le démon iSCSI. `iscsiadm` Gère les montages iSCSI et utilise la mise en réseau hôte pour communiquer avec le démon iSCSI.
IPC hôte	Le NFS utilise la communication interprocess (IPC) pour communiquer avec le NFSD.
PID de l'hôte	Nécessaire pour démarrer `rpc-statd` Pour NFS. Astra Trident interroge les processus hôte pour déterminer si `rpc-statd` Est en cours d'exécution avant le montage de volumes NFS.

Capacités	Le `SYS_ADMIN` la fonctionnalité fait partie des fonctionnalités par défaut pour les conteneurs privilégiés. Par exemple, Docker définit ces fonctionnalités pour les conteneurs privilégiés :
	`CapPrm: 0000003fffffffff`
	`CapEff: 0000003fffffffff`

Autorisations	Description
Seccomp	Le profil Seccomp est toujours « confiné » dans des conteneurs privilégiés. Par conséquent, il ne peut pas être activé sur Astra Trident.
SELinux	Sur OpenShift, des conteneurs privilégiés sont exécutés dans le `spc_t` (« conteneur super privilégié ») domaine, et les conteneurs non privilégiés sont exécutés dans le `container_t` domaine. Marche `containerd`, avec `container-selinux` tous les conteneurs sont installés dans le `spc_t` Domaine, qui désactive réellement SELinux. Astra Trident n'ajoute donc 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-…". Si cela se produit, 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 de système "Pilote FlexVolume", par conséquent, 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 aucun niveau 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 requiert la communication du IPC hôte avec `nfsd`	`true`
`hostNetwork`	Iscsiadm nécessite que le réseau hôte communique avec le démon iSCSI.	`true`
`hostPID`	Le PID 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 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 n'est pas défini `seLinuxOptions` Car il existe actuellement des différences dans le mode de gestion des conteneurs et de distribution Kubernetes de 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 requiert la communication du IPC hôte avec `nfsd`.	`true`
`allowHostNetwork`	Iscsiadm nécessite que le réseau hôte communique avec le démon iSCSI.	`true`
`allowHostPID`	Le PID 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'escalade des privilèges.	`true`
`allowPrivilegedContainer`	Les pods de nœuds Trident doivent exécuter un conteneur privilégié pour monter des volumes.	`true`
`allowedUnsafeSysctls`	Trident n'exige aucun niveau 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 n'est pas défini `seLinuxOptions` Car il existe actuellement des différences dans le mode de gestion des conteneurs et de distribution Kubernetes de 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