Référence

Trident

NetApp
July 01, 2026

Sommaire

Référence

Ports Trident

En savoir plus sur les ports que Trident utilise pour la communication.

Aperçu

Trident utilise différents ports pour communiquer au sein des clusters Kubernetes et avec les systèmes de stockage. Voici un résumé des principaux ports, de leur utilité et des considérations de sécurité associées.

- **Outbound focus** : Les nœuds Kubernetes (contrôleur et nœud de travail) initient principalement le trafic vers les LIF/IP de stockage, donc les règles iptables doivent autoriser le trafic sortant des adresses IP des nœuds vers les adresses IP de stockage spécifiques sur ces ports. Évitez les règles générales de type « any-to-any ».
- **Restrictions relatives au trafic entrant** : Limitez les ports internes de Trident au trafic interne au cluster (par exemple, en utilisant un CNI comme Calico). Aucune exposition inutile du trafic entrant sur les pare-feu hôtes.
- **Sécurité du protocole** :
  - Utilisez TCP lorsque possible (plus fiable).
  - Activez CHAP/IPsec pour iSCSI si sensible ; TLS/HTTPS pour la gestion (port 443/8443).
  - Pour NFSv4 (par défaut dans Trident), supprimez les ports UDP/plus anciens NFSv3 (par exemple, 4045-4049) si non nécessaires.
  - Limiter aux sous-réseaux de confiance ; surveiller avec des outils comme Prometheus (port 8001 optionnel).

Ports pour les nœuds de contrôleur

Ces ports sont principalement destinés à l'opérateur Trident (gestion du backend). Tous les ports internes sont au niveau du pod ; autorisez-les sur les nœuds uniquement si le pare-feu hôte interfère avec le CNI.

Port/Protocole	Direction	But	Pilote/Protocole	Notes de sécurité
TCP 8000	Entrant/Sortant (interne au cluster)	Trident REST server (communications opérateur-contrôleur)	Tous	Limiter aux CIDR de type pod ; aucune exposition externe.
TCP 8443	Entrant/Sortant (interne au cluster)	Backchannel HTTPS (API interne sécurisée)	Tous	Chiffrement TLS ; limiter au maillage de services Kubernetes si utilisé.
TCP 8001	Entrant (interne au cluster, optionnel)	Métriques Prometheus	Tous	Exposer uniquement aux outils de surveillance (par exemple, en utilisant le contrôle d'accès basé sur les rôles) ; désactiver si inutilisé.

Port/Protocole	Direction	But	Pilote/Protocole	Notes de sécurité
TCP 443	Sortant	HTTPS vers ONTAP SVM/cluster mgmt LIF	ONTAP (tous), ANF	Exiger la validation du certificat TLS ; restreindre uniquement aux adresses IP LIF de gestion.
TCP 8443	Sortant	HTTPS vers le proxy Web Services E-Series	E-Series (iSCSI)	API REST par défaut ; utilisez des certificats ; configurable dans le backend YAML.

Ports pour les nœuds de travail

Ces ports sont destinés aux ensembles de démons de nœuds CSI et aux montages de pods. Les ports de données sont sortants vers les LIF de données de stockage ; incluez les extras NFSv3 si vous utilisez NFSv3 (optionnel pour NFSv4).

Port/Protocole	Direction	But	Pilote/Protocole	Notes de sécurité
TCP 17546	Entrant (local au pod)	Sondes de liveness/readiness des nœuds CSI	Tous	Configurable (--probe-port); garantit l'absence de conflits d'hôtes; usage local uniquement.
TCP 8000	Entrant/Sortant (interne au cluster)	Serveur REST Trident	Tous	Comme ci-dessus; interne au pod.
TCP 8443	Entrant/Sortant (interne au cluster)	HTTPS de backchannel	Tous	Comme ci-dessus.
TCP 8001	Entrant (interne au cluster, optionnel)	Métriques Prometheus	Tous	Comme ci-dessus.
TCP 443	Sortant	HTTPS vers ONTAP SVM/cluster mgmt LIF	ONTAP (tous), ANF	Comme ci-dessus ; utilisé pour la découverte.
TCP 8443	Sortant	HTTPS vers le proxy Web Services E-Series	E-Series (iSCSI)	Comme ci-dessus.
TCP/UDP 111	Sortant	RPCBIND/portmapper	ONTAP-NAS (NFSv3/v4), ANF (NFS)	Requis pour v3 ; optionnel pour v4 (déchargement du pare-feu) ; à restreindre si utilisation exclusive de NFSv4.
TCP/UDP 2049	Sortant	Démon NFS	ONTAP-NAS (NFSv3/v4), ANF (NFS)	Données essentielles ; bien connues ; utilisez TCP pour la fiabilité.

Port/Protocole	Direction	But	Pilote/Protocole	Notes de sécurité
TCP/UDP 635	Sortant	Démon de montage	ONTAP-NAS (NFSv3/v4), ANF (NFS)	Montage ; rappels bidirectionnels possibles (autoriser les connexions éphémères entrantes si nécessaire).
UDP 4045	Sortant	Gestionnaire de verrous NFS (nlockmgr)	ONTAP-NAS (NFSv3)	Verrouillage de fichiers ; ignorer pour v4 (pNFS handles) ; UDP-only.
UDP 4046	Sortant	Moniteur d'état NFS (statd)	ONTAP-NAS (NFSv3)	Notifications; peut nécessiter des ports éphémères entrants (1024-65535) pour les rappels.
UDP 4049	Sortant	Démon de quota NFS (rquotad)	ONTAP-NAS (NFSv3)	Quotas; à ignorer pour la v4.
TCP 3260	Sortant	Cible iSCSI (découverte/donnée/CHAP)	ONTAP-SAN (iSCSI), E-Series (iSCSI)	Bien connu ; authentification CHAP sur ce port ; activez l'authentification CHAP mutuelle pour la sécurité.
TCP 445	Sortant	SMB/CIFS	ONTAP-NAS (SMB), ANF (SMB)	Bien connu ; utilisez SMB3 avec chiffrement (Trident annotation netapp.io/smb-encryption=true).
TCP/UDP 88 (optionnel)	Sortant	Authentification Kerberos	ONTAP (NFS/SMB/iSCSI avec Kerb)	Si Kerberos est utilisé (non par défaut) ; vers les serveurs AD, pas vers le stockage.
TCP/UDP 389 (optionnel)	Sortant	LDAP	ONTAP (NFS/SMB avec LDAP)	Similaire ; pour la résolution de noms/auth ; se limiter à AD.

Le port de la sonde de liveness/readiness peut être modifié lors de l'installation à l'aide du `--probe-port` flag. Il est important de s'assurer que ce port n'est pas utilisé par un autre processus sur les worker nodes.

API REST Trident

Bien que "commandes et options tridentctl" soit le moyen le plus simple d'interagir avec l'API REST de Trident, 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` est limité par défaut à localhost lorsqu'il s'exécute dans un pod. Pour modifier ce comportement, vous devez définir l'argument de Trident `-address` dans la configuration de son pod.

Utilisation de l'API REST

Pour des exemples de la façon dont ces API sont appelées, passez le drapeau de débogage (`-d`. Pour plus d'informations, consultez "Gérez Trident à l'aide de tridentctl".

L'API fonctionne comme suit :

GET

`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.

POST

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

Crée un objet du type spécifié.

- Nécessite une configuration JSON pour que l'objet soit créé. Pour la spécification de chaque type d'objet, reportez-vous 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'objet échoueront lors de l'opération.

SUPPRIMER

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

Supprime la ressource nommée.

> Les volumes associés aux backends ou aux classes de stockage continueront d'exister ; ils doivent être supprimés séparément. Pour plus d'informations, consultez "Gérez Trident à l'aide de tridentctl".

Options de ligne de commande

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

Journalisation

`-debug`

Active la sortie de débogage.

-loglevel <level>

Définit le niveau de journalisation (debug, info, warn, error, fatal). Par défaut, info.

Kubernetes

-k8s_pod

Utilisez cette option ou `-k8s_api_server` pour activer la prise en charge de Kubernetes. En la configurant, Trident utilise les identifiants du compte de service Kubernetes du pod qui le contient pour contacter le serveur d'API. Cela ne fonctionne que 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'elle est spécifiée, Trident se connecte au serveur d'API Kubernetes via l'adresse et le port non sécurisés fournis. Cela permet à Trident d'être déployé en dehors d'un pod ; toutefois, seules les connexions non sécurisées au serveur d'API sont prises en charge. Pour une connexion sécurisée, déployez Trident dans un pod avec l'option `-k8s_pod`.

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>

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

-config <file>

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

REST

-address <ip-or-host>

Spécifie l'adresse sur laquelle le serveur REST de Trident doit écouter. Par défaut, localhost. Lorsqu'il écoute sur localhost et s'exécute dans un pod Kubernetes, l'interface REST n'est pas directement accessible depuis l'extérieur du pod. Utilisez `-address ""` pour 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 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. La valeur par défaut est true.

Objets Kubernetes et Trident

Vous pouvez interagir avec Kubernetes et Trident via des API REST en lisant et en écrivant des objets de ressources. Plusieurs objets de ressources définissent la relation entre Kubernetes et Trident, Trident et le stockage, et Kubernetes et le stockage. Certains de ces objets sont gérés par Kubernetes et les autres sont gérés par Trident.

Comment les objets interagissent-ils entre eux ?

Peut-être que la façon la plus simple de comprendre les objets, leur utilité et la façon dont ils interagissent, est de suivre une seule demande de stockage d'un utilisateur Kubernetes :

1. Un utilisateur crée un `PersistentVolumeClaim` en demandant un nouveau `PersistentVolume` d'une taille particulière à partir d'un Kubernetes `StorageClass` qui a été préalablement configuré par l'administrateur.
2. Le Kubernetes `StorageClass` identifie Trident comme son provisionneur et inclut des paramètres qui indiquent à Trident comment provisionner un volume pour la classe demandée.
3. Trident examine son 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 le `PersistentVolume` et le stockage réel.
5. Kubernetes associe le `PersistentVolumeClaim` au nouveau `PersistentVolume`. Les Pods qui incluent le `PersistentVolumeClaim` montent ce PersistentVolume sur n'importe quel hôte sur lequel ils s'exécutent.
6. Un utilisateur crée un `VolumeSnapshot` d'un PVC existant, en utilisant un `VolumeSnapshotClass` qui pointe vers Trident.
7. Trident identifie le volume associé au PVC et crée un instantané du volume sur son backend. Il 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 impliquées dans la création d'un `PersistentVolume` et d'un `Volume`.

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

Objets `PersistentVolumeClaim` Kubernetes

Un objet `PersistentVolumeClaim`Kubernetes est 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 que vous avez définies dans la configuration du backend :

Annotation	Option de volume	Pilotes pris en charge
trident.netapp.io/fileSystem	fileSystem	ontap-san, solidfire-san, ontap-san-economy
trident.netapp.io/cloneFromPVC	cloneSourceVolume	ontap-nas, ontap-san, solidfire-san, azure-netapp-files, ontap-san-economy
trident.netapp.io/splitOnClone	splitOnClone	ontap-nas, ontap-san
trident.netapp.io/protocol	protocole	n'importe lequel
trident.netapp.io/exportPolicy	exportPolicy	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup
trident.netapp.io/snapshotPolicy	snapshotPolicy	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san
trident.netapp.io/snapshotReserve	snapshotReserve	ontap-nas, ontap-nas-flexgroup, ontap-san
trident.netapp.io/snapshotDirectory	snapshotDirectory	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup
trident.netapp.io/unixPermissions	unixPermissions	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup
trident.netapp.io/blockSize	blockSize	solidfire-san
trident.netapp.io/skipRecoveryQueue	skipRecoveryQueue	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, ontap-san-economy

Si le PV créé a la `Delete` stratégie de récupération, Trident supprime à la fois le PV et le volume sous-jacent lorsque le PV est libéré (c'est-à-dire lorsque l'utilisateur supprime le PVC). Si l'action de suppression échoue, 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 la `Retain` stratégie, Trident l'ignore et suppose que l'administrateur le supprimera de Kubernetes et du backend, 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 sous-jacent par Trident. Vous devez le supprimer à l'aide de l'API REST (`tridentctl`.

Trident prend en charge la création d'instantanés de volumes 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. Ainsi, des copies à un instant donné des PV peuvent être exposées à Kubernetes sous forme d'instantanés. Les instantanés peuvent ensuite être utilisés pour créer de nouveaux PV. Consultez `On-Demand Volume Snapshots` pour voir comment cela fonctionne.

Trident fournit également les `cloneFromPVC` et `splitOnClone` annotations pour la création de clones. Vous pouvez utiliser ces annotations pour cloner un PVC sans avoir recours à 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, telle que `trident.netapp.io/cloneFromPVC: mysql`. Avec cette annotation définie, Trident clone le volume correspondant au PVC mysql, au lieu de provisionner un volume 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 les `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` défini sur `true`, Trident sépare le volume cloné du volume parent, découplant 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 définir `trident.netapp.io/splitOnClone` ou le définir sur `false` entraîne une réduction de la consommation d'espace sur le backend, mais crée des dépendances entre les volumes parent et clone, de sorte que le volume parent ne peut pas être supprimé à moins que le clone ne le soit d'abord. Un scénario où la séparation du clone a du sens est le clonage d'un volume de base de données vide, lorsqu'on s'attend à ce que le volume et son clone divergent fortement et ne bénéficient pas des gains d'efficacité de stockage offerts par ONTAP.

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

Objets `PersistentVolume` Kubernetes

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

> Trident crée `PersistentVolume` 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 faisant référence à un volume basé sur Trident `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, il s'assure que les noms sont uniques dans leur portée.
- La taille du volume correspond au mieux à la taille demandée dans le PVC, bien qu'elle puisse être arrondie à la quantité allouable la plus proche, selon la plateforme.

Objets `StorageClass` Kubernetes

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

Les paramètres de la classe de stockage sont :

Attribut	Type	Obligatoire	Description
attributs	map[string]string	non	Voir la section attributs ci-dessous
storagePools	map[string]StringList	non	Carte des noms de backend vers des listes de pools de stockage à l'intérieur
additionalStoragePools	map[string]StringList	non	Carte des noms de backend vers des listes de pools de stockage à l'intérieur
excludeStoragePools	map[string]StringList	non	Carte des noms de backend vers des listes de pools de stockage à l'intérieur

Les attributs de stockage et leurs valeurs possibles peuvent être classés en attributs de sélection de 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	hdd, hybride, ssd	Le pool contient des médias de ce type ; hybride signifie les deux	Type de média spécifié	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, solidfire-san

Attribut	Type	Valeurs	Offre	Demande	Soutenu par
provisioningType	chaîne	mince, épais	Pool prend en charge cette méthode de provisionnement	Méthode de provisionnement spécifiée	épais : all ontap ; mince : all ontap & solidfire-san
backendType	chaîne	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, solidfire-san, azure-netapp-files, ontap-san-economy	Pool appartient à ce type de backend	Backend spécifié	Tous les drivers
instantanés	bool	vrai, faux	Pool prend en charge les volumes avec snapshots	Volume avec instantanés activés	ontap-nas, ontap-san, solidfire-san
clones	bool	vrai, faux	Pool prend en charge le clonage des volumes	Volume avec clones activés	ontap-nas, ontap-san, solidfire-san
chiffrement	bool	vrai, faux	Pool prend en charge les volumes chiffrés	Volume avec chiffrement activé	ontap-nas, ontap-nas-economy, ontap-nas-flexgroups, ontap-san
Op E/S par sec	int	entier positif	Pool est capable de garantir des IOPS dans cette plage	Volume garanti ces IOPS	solidfire-san

[1]: Non pris en charge par ONTAP Select

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

Idéalement, vous pouvez utiliser `attributes` seul pour modéliser les qualités du stockage dont vous avez besoin pour 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 n'arrivez pas à utiliser `attributes` pour sélectionner automatiquement les bons pools pour une classe, vous pouvez utiliser les paramètres `storagePools` et `additionalStoragePools` pour affiner davantage les pools ou même sélectionner un ensemble spécifique de pools.

Vous pouvez utiliser le `storagePools` paramètre pour restreindre davantage l'ensemble des pools

correspondant à tout `attributes` spécifié. Autrement dit, Trident utilise l'intersection des pools identifiés par les paramètres `attributes` et `storagePools` pour le provisionnement. Vous pouvez utiliser chaque paramètre individuellement ou les deux ensemble.

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

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

Dans les `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 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 des valeurs regex pour le backend et les valeurs de la liste. Vous pouvez utiliser `tridentctl get backend` pour obtenir la liste des backends et de leurs pools.

Attributs Kubernetes

Ces attributs n'ont aucune incidence sur la sélection des pools de stockage/backends par Trident lors du provisionnement dynamique. Au lieu de cela, 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	Pilotes 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
allowVolumeExpansion	booléen	vrai, faux	Activer ou désactiver la prise en charge de l'agrandissement de la taille du PVC	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, ontap-san-economy, solidfire-san, azure-netapp-files	1.11+
volumeBindingMode	chaîne	Immédiat, WaitForFirstConsumer	Choisissez quand la liaison de volume et le provisionnement dynamique ont lieu	Tous	1.19 - 1.26

- Le `fsType` paramètre est utilisé pour 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 qu'un système de fichiers existe. La propriété du volume peut être contrôlée à l'aide du `fsGroup` contexte de sécurité d'un pod uniquement si `fsType` est défini. Consultez "Kubernetes : configurer un contexte de sécurité pour un pod ou un conteneur" pour un aperçu de la configuration de la propriété du volume à l'aide du `fsGroup` contexte. Kubernetes appliquera la valeur `fsGroup` 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 l'exportation NFS. Pour utiliser `fsGroup` la classe de stockage, il faut toujours spécifier un `fsType`. Vous pouvez le définir sur `nfs` ou sur toute valeur non nulle.

- Reportez-vous à "Étendre les volumes" pour plus de détails sur l'expansion du volume.
- Le bundle d'installation Trident fournit 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.

Objets `VolumeSnapshotClass` Kubernetes

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

A `VolumeSnapshotClass` doit être défini par un administrateur afin de créer des snapshots. Une classe de snapshot 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 de la classe `csi-snapclass` sont gérées par Trident. Le `deletionPolicy` spécifie l'action à entreprendre lorsqu'un snapshot doit être supprimé. Lorsque `deletionPolicy` est défini sur `Delete`, les objets snapshot de volume ainsi que le snapshot sous-jacent sur le cluster de stockage sont supprimés lorsqu'un snapshot est supprimé. Sinon, le définir sur `Retain` signifie que `VolumeSnapshotContent` et le snapshot physique sont conservés.

Objets `VolumeSnapshot` Kubernetes

Un objet Kubernetes `VolumeSnapshot` est une demande de création d'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 backend et l'expose en créant un objet `VolumeSnapshotContent` unique. Vous pouvez créer des snapshots à partir de PVC existants et utiliser les snapshots comme DataSource 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 snapshots, Trident marque le volume sous-jacent de ce PVC comme étant en **Deleting**, mais ne le supprime pas complètement. Le volume est supprimé lorsque tous les snapshots associés sont supprimés.

Objets `VolumeSnapshotContent` Kubernetes

Un objet Kubernetes `VolumeSnapshotContent` 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. À l'instar de `PersistentVolumeClaim` et de `PersistentVolume` objets, lorsqu'un instantané est créé, l'objet `VolumeSnapshotContent` maintient une correspondance un-à-un avec l'objet `VolumeSnapshot` qui a demandé la création de l'instantané.

L'`VolumeSnapshotContent`objet contient des détails qui identifient de manière unique l'instantané, tels que le `snapshotHandle`. Cet `snapshotHandle`ensemble est une combinaison unique du nom du PV et du nom de l' `VolumeSnapshotContent`objet.

Lorsqu'une requête de snapshot est reçue, Trident crée le snapshot sur le backend. Une fois le snapshot créé, Trident configure un `VolumeSnapshotContent` objet et expose ainsi le snapshot à l'API Kubernetes.

> ⓘ En règle générale, vous n'avez pas besoin de gérer l' `VolumeSnapshotContent`objet. Une exception à cela est lorsque vous souhaitez "importer un instantané de volume"créer en dehors de Trident.

Objets `VolumeGroupSnapshotClass` Kubernetes

Les objets Kubernetes `VolumeGroupSnapshotClass` sont analogues à `VolumeSnapshotClass`. Ils permettent de définir plusieurs classes de stockage et sont référencés par les snapshots de groupes de volumes afin d'associer le snapshot à la classe de snapshot requise. Chaque snapshot de groupe de volumes est associé à une seule classe de snapshot de groupe de volumes.

A `VolumeGroupSnapshotClass` doit être défini par un administrateur afin de créer 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 de la classe `csi-group-snap-class` sont gérées par Trident. Le `deletionPolicy` spécifie l'action à entreprendre lorsqu'un snapshot de groupe doit être supprimé. Lorsque `deletionPolicy` est défini sur `Delete`, les objets snapshot de groupe de volumes ainsi que le snapshot sous-jacent sur le cluster de stockage sont supprimés lors de la suppression d'un snapshot. Sinon, le définir sur `Retain` signifie que `VolumeGroupSnapshotContent` et le snapshot physique sont conservés.

Objets `VolumeGroupSnapshot` Kubernetes

Un objet Kubernetes `VolumeGroupSnapshot` correspond à une requête de création d'un instantané de plusieurs volumes. De même qu'un PVC représente une requête d'un utilisateur pour un volume, un instantané de groupe de volumes correspond à une requête d'un utilisateur visant à 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 backend et expose le snapshot en créant un objet `VolumeGroupSnapshotContent` unique. Vous pouvez créer des snapshots à partir de PVC existants et utiliser les snapshots comme DataSource 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 snapshots, Trident marque le volume sous-jacent de ce PVC comme étant en **Deleting**, mais ne le supprime pas complètement. Le snapshot du groupe de volumes est supprimé lorsque tous les snapshots associés sont supprimés.

Objets `VolumeGroupSnapshotContent` Kubernetes

Un objet Kubernetes `VolumeGroupSnapshotContent` 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. À l'instar de `PersistentVolumeClaim` et de `PersistentVolume` objets, lorsqu'un instantané est créé, l'objet `VolumeSnapshotContent` maintient une correspondance un-à-un avec l'objet `VolumeSnapshot` qui a demandé la création de l'instantané.

L `VolumeGroupSnapshotContent`objet contient des détails qui identifient le groupe d'instantanés, tels que le `volumeGroupSnapshotHandle` et les volumeSnapshotHandles individuels existant sur le système de stockage.

Lorsqu'une requête de snapshot est reçue, Trident crée le snapshot du groupe de volumes sur le backend. Après la création du snapshot du groupe de volumes, Trident configure un `VolumeGroupSnapshotContent` objet et expose ainsi le snapshot à l'API Kubernetes.

Objets `CustomResourceDefinition` Kubernetes

Les ressources personnalisées Kubernetes sont des points de terminaison de l'API Kubernetes définis par l'administrateur et utilisées 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 système de métadonnées. Cela élimine le besoin d'un système de stockage séparé pour Trident.

Trident utilise `CustomResourceDefinition` des objets pour 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 framework de snapshots de volumes CSI introduit certains CRD nécessaires pour définir les snapshots de volumes.

Les CRD sont un concept Kubernetes. Les objets des ressources définies ci-dessus sont créés par Trident. À titre d'exemple simple, lorsqu'un backend est créé à l'aide de `tridentctl`, un objet CRD correspondant `tridentbackends` est créé pour être utilisé par Kubernetes.

Voici quelques points à retenir concernant les CRD de Trident :

- Lorsque Trident est installé, 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 la commande `tridentctl uninstall`, les pods Trident sont supprimés, mais les CRD créées ne sont pas nettoyées. Consultez "Désinstaller Trident" pour comprendre comment Trident peut être complètement supprimé et reconfiguré à partir de zéro.

Trident `StorageClass` objets

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

> Avec Kubernetes, ces objets sont créés automatiquement lorsqu'une instance Kubernetes `StorageClass` qui utilise Trident comme provisionneur est enregistrée.

Les classes de stockage définissent un ensemble d'exigences pour les volumes. Trident associe 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 des classes de stockage en utilisant l'API REST. Cependant, pour les déploiements Kubernetes, nous nous attendons à ce qu'elles soient créées lors de l'enregistrement de nouveaux objets `StorageClass` Kubernetes.

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 l'objet `StorageClass` Kubernetes.

Pour plus d'informations sur la manière de construire ces objets, reportez-vous à "configuration des backends".

Trident `StoragePool` objets

Les pools de stockage représentent les emplacements distincts disponibles pour le provisionnement sur chaque backend. Pour ONTAP, ils correspondent à des agrégats dans les SVM. Pour NetApp HCI/SolidFire, ils correspondent à des niveaux de QoS définis par l'administrateur. Chaque pool de stockage possède un ensemble d'attributs de stockage spécifiques, qui définissent ses caractéristiques de performance et 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 constituent l'unité de base du provisionnement et comprennent les points de terminaison backend, tels que les partages NFS, ainsi que les LUN iSCSI et FC. Dans Kubernetes, ils 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 provisionné.
- Lors de la suppression d'un PV avec des snapshots associés, le volume Trident correspondant passe à l'état **Deleting**. 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
storageClass	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 : "file" ou "block"
internalName	chaîne	non	Nom de l'objet sur le système de stockage ; généré par Trident
cloneSourceVolume	chaîne	non	ontap (nas, san) & solidfire-*: Nom du volume à cloner
splitOnClone	chaîne	non	ontap (nas, san): Séparer le clone de son parent
snapshotPolicy	chaîne	non	ontap-*: Stratégie d'instantané à utiliser
snapshotReserve	chaîne	non	ontap-*: Pourcentage du volume réservé aux instantanés
exportPolicy	chaîne	non	ontap-nas* : Politique d'export à utiliser

Attribut	Type	Obligatoire	Description
snapshotDirectory	bool	non	ontap-nas* : Whether the snapshot directory is visible
unixPermissions	chaîne	non	ontap-nas*: Permissions UNIX initiales
blockSize	chaîne	non	solidfire-*: Taille du bloc/secteur
fileSystem	chaîne	non	Type de système de fichiers
skipRecoveryQueue	chaîne	non	Lors de la suppression d'un volume, ignorez la file d'attente de récupération dans le stockage et supprimez le volume immédiatement.

Trident génère `internalName` lors de la création du volume. Cela consiste en deux étapes. Tout d'abord, il ajoute le préfixe de stockage (soit le préfixe par défaut `trident` soit 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 normalisation du nom, en remplaçant les caractères non autorisés dans le backend. 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 traits de soulignement 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 nous attendons à ce que la plupart des utilisateurs utilisent la méthode standard Kubernetes `PersistentVolumeClaim`. Trident crée cet objet volume automatiquement dans le cadre du processus de provisionnement.

Trident `Snapshot` objets

Les snapshots sont des copies de volumes à un instant précis, permettant de provisionner de nouveaux volumes ou de restaurer un état. Dans Kubernetes, ils correspondent directement à `VolumeSnapshotContent` objets. Chaque snapshot est associé à un volume, qui constitue la source des données du snapshot.

Chaque `Snapshot` 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
internalName	Chaîne	Oui	Nom de l'objet snapshot Trident sur le système de stockage

Attribut	Type	Obligatoire	Description
volumeName	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 provisionné.

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

Trident `ResourceQuota` objet

Le daemonset Trident consomme une `system-node-critical`Priority Class — la Priority Class 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 afin de garantir que la classe de priorité « system-node-critical » sur le daemonset Trident soit respectée. Avant le déploiement et la création du daemonset, Trident recherche l' `ResourceQuota` objet et, s'il n'est pas trouvé, l'applique.

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

L'exemple suivant montre un objet `ResourceQuota` qui donne la priorité au 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".

Nettoyez `ResourceQuota` si l'installation échoue

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

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

Retirer `ResourceQuota`

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

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

Normes de sécurité des pods (PSS) et Security Context Constraints (SCC)

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

PSS remplace les Pod Security Policies (PSP). PSP a été déprécié dans Kubernetes v1.21 et sera supprimé dans la v1.25. Pour plus d'informations, consultez "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 Trident du nœud doit exécuter un conteneur privilégié. Pour plus d'informations, consultez "Kubernetes : propagation du montage".
Réseau hôte	Nécessaire 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` est en cours d'exécution avant de monter les volumes NFS.

Autorisation	Description

Capacités	La `SYS_ADMIN` fonctionnalité est fournie dans le cadre des fonctionnalités par défaut pour les conteneurs privilégiés. Par exemple, Docker définit ces fonctionnalités 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 s'exécutent dans le domaine `spc_t` (« Super Privileged Container »), et les conteneurs non privilégiés s'exécutent dans le domaine `container_t`. Sur `containerd`, avec `container-selinux` installé, tous les conteneurs s'exécutent dans le domaine `spc_t`, 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 pods (PSS)

Étiquette	Description	Défaut
`pod-security.kubernetes.io/enforce` `pod-security.kubernetes.io/enforce-version`	Autorise le contrôleur Trident et les nœuds à être admis 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 « Error creating: … » ou un « Warning: trident-csi-… ». Si cela se produit, vérifiez si l'étiquette d'espace de noms pour `privileged` a été modifiée. 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 CSI éphémères en ligne.	Vide

Champ	Description	Défaut
`allowedCapabilities`	Les conteneurs Trident non privilégiés ne nécessitent pas plus de capacités que l'ensemble par défaut et 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 Trident du nœud 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 opération non sécurisée `sysctls`.	Vide
`defaultAddCapabilities`	Aucune capacité n'est requise pour être ajoutée aux conteneurs privilégiés.	Vide
`defaultAllowPrivilegeEscalation`	L'autorisation d'élévation de privilèges est gérée dans chaque pod Trident.	`false`
`forbiddenSysctls`	Aucun `sysctls` n'est autorisé.	Vide
`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`	L'identifiant du processus hôte (PID) 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`

Champ	Description	Défaut
`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 configure 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 pods Trident nécessitent ces plugins de volume.	`hostPath, projected, emptyDir`

Security Context Constraints (SCC)

Étiquettes	Description	Défaut
`allowHostDirVolumePlugin`	Les pods de nœud Trident montent le système de fichiers racine du nœud.	`true`
`allowHostIPC`	Le montage des 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`	L'identifiant du processus hôte (PID) 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`

Étiquettes	Description	Défaut
`allowPrivilegedContainer`	Les pods de nœuds Trident doivent exécuter un conteneur privilégié pour pouvoir monter des volumes.	`true`
`allowedUnsafeSysctls`	Trident ne nécessite aucune opération non sécurisée `sysctls`.	`none`
`allowedCapabilities`	Les conteneurs Trident non privilégiés ne nécessitent pas plus de capacités que l'ensemble par défaut et les conteneurs privilégiés se voient accorder toutes les capacités possibles.	Vide
`defaultAddCapabilities`	Aucune capacité 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 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`
`seLinuxContext`	Trident ne configure 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 "Unconfined".	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.	s.o.
`volumes`	Les pods Trident nécessitent ces plugins de volume.	`hostPath, downwardAPI, projected, emptyDir`

Informations sur le copyright

Informations sur les marques commerciales