Documentation d’Astra Trident 23.01.1

Astra Trident

NetApp
July 17, 2025

This PDF was generated from https://docs.netapp.com/fr-fr/trident-2301/index.html on July 17, 2025.
Always check docs.netapp.com for the latest.

Sommaire

Documentation d'Astra Trident 23.01.1 . . . 1
Notes de version . . . 2
Nouveautés de la version 23.01.1 . . . 2
Correctifs . . . 2
Changements en 23.01 . . . 2
Correctifs . . . 2
Améliorations . . . 2
Dérations . . . 3
Changements en 22.10 . . . 3
Correctifs . . . 3
Améliorations . . . 4
Dérations . . . 4
Changements en 22.07 . . . 4
Correctifs . . . 4
Améliorations . . . 4
Dérations . . . 5
Suppressions . . . 5
Documentation . . . 5
Changements en 22.04 . . . 5
Correctifs . . . 5
Améliorations . . . 6
Suppressions . . . 6
Changements en 22.01.1 . . . 6
Correctifs . . . 6
Changements en 22.01.0 . . . 6
Correctifs . . . 6
Améliorations . . . 6
Dérations . . . 7
Changements en 21.10.1 . . . 7
Correctifs . . . 7
Changements en 21.10.0 . . . 7
Correctifs . . . 7
Améliorations . . . 8
Améliorations expérimentales . . . 8
Problèmes connus . . . 8
Trouvez plus d'informations . . . 9
Concepts . . . 10
Découvrez Astra Trident . . . 10
Présentation . . . 10
Architectures de cluster Kubernetes prises en charge . . . 10
Qu'est-ce qu'Astra ? . . . 10
Pour en savoir plus . . . 11
Pilotes ONTAP . . . 11

Découvrez les pilotes de stockage ONTAP 11
Provisionnement 12
Association de classe de stockage 12
Création du volume 13
Snapshots de volume 13
En savoir plus sur la création de snapshots de volume 13
Pools virtuels 14
En savoir plus sur les pools virtuels 14
Groupes d'accès de volume 15
En savoir plus sur les groupes d'accès aux volumes 15
Commencez 17
Essayez-le 17
Découvrez Test Drive 17
De formation 17
Découvrez Astra Trident 23.01, un document essentiel 17
Systèmes front-end (orchestrateurs) pris en charge 17
Systèmes back-end pris en charge (stockage) 18
Configuration requise 18
Systèmes d'exploitation hôtes testés 19
Configuration de l'hôte 19
Configuration du système de stockage 19
Ports Trident d'Astra 19
Images de conteneur et versions Kubernetes correspondantes 20
Installer Astra Trident 22
Découvrez l'installation d'Astra Trident 22
Installer à l'aide de l'opérateur Trident 26
Installation à l'aide de tridentctl 51
Et la suite ? 55
Étape 1 : créer un back-end 55
Étape 2 : créer une classe de stockage 56
Étape 3 : provisionner le premier volume 58
Étape 4 : montez les volumes dans un pod 59
Gérez Astra Trident 61
Mettez à niveau Astra Trident 61
Mettez à niveau Astra Trident 61
Mise à niveau avec l'opérateur 62
Mise à niveau avec tridentctl 70
Désinstaller Astra Trident 74
Désinstaller en utilisant Helm 74
Désinstaller en utilisant l'opérateur Trident 75
Désinstaller à l'aide de `tridentctl` 76
Revenir à la version antérieure d'Astra Trident 76
Quand revenir à une version antérieure 76
Si vous ne devez pas revenir à une version antérieure 76
Procédure de rétrogradation lorsque Astra Trident est installé à l'aide de l'opérateur 76

Processus de rétrogradation lors de l'installation d'Astra Trident à l'aide de `tridentctl` 78
Avec Astra Trident 80
Préparez le nœud de travail 80
Choisir les bons outils 80
Détection des services de nœud 80
Volumes NFS 80
Volumes iSCSI 81
Configuration des systèmes back-end 84
Azure NetApp Files 85
Configurer un système Cloud Volumes Service pour Google Cloud backend 97
Configurer un système NetApp HCI ou SolidFire backend 113
Configurer un système back-end avec les pilotes SAN ONTAP 120
Configurer un système NAS backend ONTAP 142
Amazon FSX pour NetApp ONTAP 168
Création de systèmes back-end avec kubectl 180
`TridentBackendConfig` 180
Présentation des étapes 181
Étape 1 : créez un code secret Kubernetes 182
Étape 2 : créez le `TridentBackendConfig` CR 183
Étape 3 : vérifier l'état du `TridentBackendConfig` CR 184
(Facultatif) étape 4 : pour plus de détails 184
Effectuer la gestion back-end avec kubectl 186
Supprimer un back-end 186
Affichez les systèmes back-end existants 186
Mettre à jour un back-end 186
Gestion back-end avec tridentctl 187
Créer un back-end 187
Supprimer un back-end 187
Affichez les systèmes back-end existants 188
Mettre à jour un back-end 188
Identifier les classes de stockage qui utilisent un système back-end 188
Passez d'une option de gestion back-end à une autre 189
Gérez `tridentctl` utilisation de systèmes back-end `TridentBackendConfig` 189
Gérez `TridentBackendConfig` utilisation de systèmes back-end `tridentctl` 193
Gérer les classes de stockage 195
Concevez une classe de stockage 195
Créer une classe de stockage 195
Supprimer une classe de stockage 195
Afficher les classes de stockage existantes 196
Définir une classe de stockage par défaut 196
Identifier le système back-end pour une classe de stockage 197
Réaliser des opérations de volume 197
Utiliser la topologie CSI 197
Travailler avec des instantanés 205

Développement des volumes . . . 209
Importer des volumes . . . 216
Partager un volume NFS entre les espaces de noms . . . 222
Caractéristiques . . . 222
Démarrage rapide . . . 223
Configurer les espaces de noms source et de destination . . . 224
Supprimer un volume partagé . . . 225
Utiliser `tridentctl get` pour interroger des volumes subordonnés . . . 225
Limites . . . 226
Pour en savoir plus . . . 226
Contrôle d'Astra Trident . . . 226
Étape 1 : définir une cible Prometheus . . . 227
Étape 2 : créer un ServiceMonitor Prometheus . . . 227
Étape 3 : interroger les mesures Trident avec PromQL . . . 228
Découvrez la télémétrie AutoSupport d'Astra Trident . . . 229
Désactivation des metrics d'Astra Trident . . . 230
Astra Trident pour Docker . . . 231
Conditions préalables au déploiement . . . 231
Vérifier les exigences . . . 231
Déployez Astra Trident . . . 234
Méthode de plug-in géré Docker (version 1.13/17.03 et ultérieure) . . . 234
Méthode traditionnelle (version 1.12 ou antérieure) . . . 236
Commencez avec Astra Trident au démarrage du système . . . 237
Mise à niveau ou désinstallation d'Astra Trident . . . 238
Mise à niveau . . . 239
Désinstaller . . . 240
Utilisation de volumes . . . 240
Créer un volume . . . 240
Supprimer un volume . . . 241
Clonez un volume . . . 241
Accéder aux volumes créés en externe . . . 243
Options de volume spécifiques au conducteur . . . 243
Collecte des journaux . . . 248
Collecte des journaux pour le dépannage . . . 248
Conseils généraux de dépannage . . . 249
Gérez plusieurs instances Trident d'Astra . . . 249
Étapes du plug-in géré par Docker (version 1.13/17.03 ou ultérieure) . . . 249
Étapes pour les versions traditionnelles (version 1.12 ou antérieure) . . . 250
Options de configuration du stockage . . . 250
Options de configuration globale . . . 250
Configuration ONTAP . . . 251
Configuration logicielle Element . . . 257
Problèmes et limites connus . . . 259
La mise à niveau de Trident Docker Volume Plug-in vers la version 20.10 et ultérieure à partir des versions plus anciennes entraîne un échec de mise à niveau, sans erreur de fichier ou de répertoire

de ce type. . . . 259
Les noms de volumes doivent comporter au moins 2 caractères. . . . 260
Docker Swarm dispose de certains comportements qui empêchent Astra Trident de les prendre en charge avec chaque combinaison de stockage et de pilote. . . . 260
Si un FlexGroup est provisionné, ONTAP ne provisionne pas un deuxième FlexGroup si le deuxième FlexGroup dispose d'un ou de plusieurs agrégats en commun avec la FlexGroup provisionnée. . . . 260
Foire aux questions . . . 261
Questions générales . . . 261
À quelle fréquence Astra Trident est-il commercialisé ? . . . 261
Est-ce que l'Astra Trident prend en charge toutes les fonctionnalités publiées dans une version particulière de Kubernetes ? . . . 261
Est-ce que l'Astra Trident se base sur d'autres produits NetApp pour son fonctionnement ? . . . 261
Comment obtenir les informations complètes sur la configuration d'Astra Trident ? . . . 261
Puis-je obtenir des metrics sur le provisionnement du stockage par Astra Trident ? . . . 261
L'expérience utilisateur change-t-elle lors de l'utilisation d'Astra Trident en tant que provisionnement CSI ? . . . 261
Installez et utilisez Astra Trident sur un cluster Kubernetes . . . 261
Quelles sont les versions prises en charge de `etcd`? . . . 262
Est-ce que Astra Trident prend en charge une installation hors ligne à partir d'un registre privé ? . . . 262
Puis-je installer Astra Trident à distance ? . . . 262
Puis-je configurer la haute disponibilité avec Astra Trident ? . . . 262
Astra Trident a-t-il besoin d'accéder au namespace du système kube ? . . . 262
Quels sont les rôles et privilèges utilisés par Astra Trident ? . . . 262
Est-il possible de générer localement les fichiers de manifeste exacts qu'Astra Trident utilise pour l'installation ? . . . 262
Puis-je partager le même SVM back-end ONTAP pour deux instances Astra Trident distinctes pour deux clusters Kubernetes distincts ? . . . 262
Est-il possible d'installer Astra Trident sous ContainerLinux (anciennement CoreOS) ? . . . 263
Puis-je utiliser Astra Trident avec NetApp Cloud Volumes ONTAP ? . . . 263
Astra Trident fonctionne-t-il avec NetApp Cloud volumes Services ? . . . 263
Dépannage et support . . . 263
NetApp prend-il en charge Astra Trident ? . . . 263
Comment puis-je soulever un dossier de demande de support ? . . . 263
Comment générer un bundle de journaux de support ? . . . 263
Que faire si j'ai besoin de demander une nouvelle fonctionnalité ? . . . 263
Où puis-je soulever un défaut ? . . . 263
Que se passe-t-il si j'ai une brève question sur Astra Trident et que j'ai besoin de précisions ? Y a-t-il une communauté ou un forum? . . . 263
Le mot de passe de mon système de stockage a changé et Astra Trident ne fonctionne plus, comment récupérer? . . . 264
Astra Trident ne trouve pas le nœud Kubernetes. Comment résoudre ce problème ? . . . 264
Si le pod Trident est détruit, ces données seront-elles perdues ? . . . 264
Mettez à niveau Astra Trident . . . 264
Est-il possible de mettre à niveau une version plus ancienne directement vers une version plus récente (sans passer par quelques versions) ? . . . 264

Est-il possible de revenir à une version antérieure de Trident ? . . . 264
Gestion des systèmes back-end et des volumes . . . 264
Dois-je définir à la fois des LIF de données et de gestion dans un fichier de définition du back-end ONTAP ? . . . 264
L'Astra Trident peut-il configurer le protocole CHAP pour les systèmes back-end ONTAP ? . . . 265
Comment gérer les règles d'exportation avec Astra Trident ? . . . 265
Pouvons-nous spécifier un port dans le DataLIF ? . . . 265
Les adresses IPv6 peuvent-elles être utilisées pour les LIF de données et de gestion ? . . . 265
Est-il possible de mettre à jour la LIF de gestion en back-end ? . . . 265
Est-il possible de mettre à jour la LIF de données sur le backend ? . . . 265
Est-il possible de créer plusieurs systèmes back-end dans Astra Trident pour Kubernetes ? . . . 265
Comment Astra Trident stocke-t-il les identifiants back-end ? . . . 266
Comment l'Astra Trident sélectionne-t-il un système back-end spécifique ? . . . 266
Comment s'assurer qu'Astra Trident ne provisionne pas d'un back-end spécifique ? . . . 266
Si plusieurs systèmes back-end sont de même type, comment Astra Trident sélectionne-il le back-end à utiliser ? . . . 266
Astra Trident prend-il en charge le protocole CHAP bidirectionnel avec Element/SolidFire ? . . . 266
Comment Astra Trident déploie-t-il des qtrees sur un volume ONTAP ? Combien de qtrees peuvent-ils être déployés sur un seul volume ? . . . 266
Comment définir des autorisations Unix pour les volumes provisionnés sur ONTAP NAS ? . . . 266
Comment configurer un ensemble explicite d'options de montage NFS ONTAP lors du provisionnement d'un volume ? . . . 266
Comment définir les volumes provisionnés sur une export policy spécifique ? . . . 267
Comment définir le chiffrement de volume avec Astra Trident et ONTAP ? . . . 267
Quelle est la meilleure façon d'implémenter la QoS pour ONTAP avec Astra Trident ? . . . 267
Comment puis-je spécifier le provisionnement fin ou non fin avec Astra Trident ? . . . 267
Comment puis-je m'assurer que les volumes utilisés ne sont pas supprimés même si je supprime accidentellement le volume de volume persistant ? . . . 267
Puis-je augmenter les demandes de volume persistant NFS créées par Astra Trident ? . . . 267
Si je dispose d'un volume créé en dehors d'Astra Trident, est-il possible de l'importer dans Astra Trident ? . . . 267
Puis-je importer un volume en mode SnapMirror Data protection (DP) ou hors ligne ? . . . 267
Puis-je étendre les demandes de volume persistant iSCSI créées par Astra Trident ? . . . 268
Comment un quota de ressources est-il traduit-il vers un cluster NetApp ? . . . 268
Est-il possible de créer des copies Snapshot de volume avec Astra Trident ? . . . 268
Quels sont les pilotes qui prennent en charge les copies Snapshot de volume Astra Trident ? . . . 268
Comment effectuer une sauvegarde Snapshot d'un volume provisionné par Astra Trident avec ONTAP ? . . . 268
Puis-je définir un pourcentage de réserve de snapshot pour un volume provisionné via Astra Trident ? 268
Puis-je accéder directement au répertoire de snapshot de volume et copier les fichiers ? . . . 269
Puis-je configurer SnapMirror pour des volumes avec Astra Trident ? . . . 269
Comment restaurer des volumes persistants à un snapshot ONTAP spécifique ? . . . 269
Trident peut-il provisionner des volumes sur des SVM dont un miroir de partage de charge est configuré ? . . . 269
Comment puis-je séparer l'utilisation de la classe de stockage pour chaque client/locataire ? . . . 269

Assistance . . . 270
Dépannage . . . 271
Dépannage général . . . 271
Dépannage d'un échec de déploiement Trident à l'aide de l'opérateur . . . 273
Dépannage d'un déploiement Trident non réussi à l'aide de `tridentctl` . . . 275
Meilleures pratiques et recommandations . . . 276
Déploiement . . . 276
Déploiement dans un namespace dédié . . . 276
Utilisez les quotas et les limites des plages pour contrôler la consommation du stockage . . . 276
Configuration de stockage sous-jacente . . . 276
Présentation de la plateforme . . . 276
Meilleures pratiques pour ONTAP et Cloud Volumes ONTAP . . . 276
Les meilleures pratiques pour SolidFire . . . 281
Où trouver plus d'informations ? . . . 283
Intégrez Astra Trident . . . 283
Choix et déploiement du conducteur . . . 283
Conception de classe de stockage . . . 287
Conception de pool virtuel . . . 288
Opérations de volume . . . 289
Le déploiement des services OpenShift . . . 291
Services de metrics . . . 293
Protection des données . . . 294
Sauvegardez le `etcd` données de cluster . . . 294
Récupérer les données à l'aide des snapshots ONTAP . . . 295
Réplication des données à l'aide de ONTAP . . . 296
Restaurez les données à l'aide des snapshots Element . . . 299
Sécurité . . . 299
Sécurité . . . 299
Configuration de clé unifiée Linux (LUKS) . . . 301
Référence . . . 306
Ports Trident d'Astra . . . 306
Ports Trident d'Astra . . . 306
API REST d'Astra Trident . . . 306
Quand utiliser l'API REST . . . 306
Avec l'API REST . . . 306
Options de ligne de commande . . . 307
Journalisation . . . 307
Kubernetes . . . 307
Docker . . . 307
REPOS . . . 308
Produits NetApp intégrés avec Kubernetes . . . 308
Astra . . . 308
ONTAP . . . 308
Cloud Volumes ONTAP . . . 309
Amazon FSX pour NetApp ONTAP . . . 309

Logiciel Element . . . 309
NetApp HCI . . . 309
Azure NetApp Files . . . 309
Cloud Volumes Service pour Google Cloud . . . 309
Kubernetes et objets Trident . . . 309
Comment les objets interagissent-ils les uns avec les autres ? . . . 310
Kubernetes `PersistentVolumeClaim` objets . . . 310
Kubernetes `PersistentVolume` objets . . . 312
Kubernetes `StorageClass` objets . . . 312
Kubernetes `VolumeSnapshotClass` objets . . . 316
Kubernetes `VolumeSnapshot` objets . . . 317
Kubernetes `VolumeSnapshotContent` objets . . . 317
Kubernetes `CustomResourceDefinition` objets . . . 317
Trident `StorageClass` objets . . . 318
Objets back-end Trident . . . 318
Trident `StoragePool` objets . . . 319
Trident `Volume` objets . . . 319
Trident `Snapshot` objets . . . 320
Astra Trident `ResourceQuota` objet . . . 321
commandes et options tridentctl . . . 322
Commandes et options disponibles . . . 322
`create` . . . 323
`delete` . . . 324
`get` . . . 324
`images` . . . 324
`import volume` . . . 325
`install` . . . 325
`logs` . . . 326
`send` . . . 326
`uninstall` . . . 327
`update` . . . 327
`upgrade` . . . 327
`version` . . . 327
Normes de sécurité de pod (PSS) et contraintes de contexte de sécurité (SCC) . . . 328
Contexte de sécurité Kubernetes requis et champs associés . . . 328
Normes de sécurité du pod (PSS) . . . 329
Politiques de sécurité des pods (PSP) . . . 329
Contraintes de contexte de sécurité (SCC) . . . 331
Versions antérieures de la documentation . . . 333
Mentions légales . . . 334
Droits d’auteur . . . 334
Marques déposées . . . 334

Brevets 334
Politique de confidentialité 334
Source ouverte 334

Documentation d’Astra Trident 23.01.1

Notes de version

Dans les notes de version, vous trouverez des informations sur les nouvelles fonctionnalités, les améliorations et les correctifs de bogues de la dernière version d'Astra Trident.

> Le `tridentctl` Binaire pour Linux fourni dans le fichier zip du programme d'installation est la version testée et prise en charge. Sachez que le `macos` binaire fourni dans le `/extras` une partie du fichier zip n'est pas testée ou prise en charge.

Nouveautés de la version 23.01.1

Correctifs

- Correction de l'opérateur Trident pour utiliser IPv6 localhost pour l'installation lorsqu'il est spécifié dans spec.
- Correction des autorisations de rôle de cluster opérateur Trident synchronisées avec les autorisations de bundle "Question no 799".
- Ajout d'un correctif pour permettre l'exécution des processus externes.
- Résolution du problème de connexion d'un volume de bloc brut sur plusieurs nœuds en mode RWX.
- Prise en charge du clonage FlexGroup fixe et importation de volumes pour les volumes SMB.

Changements en 23.01

> Kubernetes 1.26 est désormais pris en charge par Trident. Veuillez mettre à niveau Astra Trident avant de mettre à niveau Kubernetes.

Correctifs

- Kubernetes : ajout d'options pour exclure la création de règles de sécurité du Pod pour réparer les installations Trident via Helm ("Questions #783, #794").

Améliorations

Kubernetes

- Prise en charge ajoutée de Kubernetes 1.26.
- Amélioration de l'utilisation globale des ressources RBAC Trident ("Numéro 757").
- Automatisation ajoutée pour détecter et corriger les sessions iSCSI interrompues ou obsolètes sur les nœuds hôtes.
- Ajout de la prise en charge de l'extension des volumes chiffrés LUKS.
- Kubernetes : ajout de la prise en charge de la rotation des identifiants pour les volumes chiffrés LUKS.

Astra Trident

- Ajout de la prise en charge des volumes SMB avec Amazon FSX pour ONTAP au pilote de stockage ontap-nas.

- Ajout de la prise en charge des autorisations NTFS lors de l'utilisation de volumes SMB.
- Ajout de la prise en charge des pools de stockage pour les volumes GCP avec le niveau de service CVS.
- Ajout de la prise en charge de l'utilisation facultative de flexgroupAgrégateList lors de la création de FlexGroups avec le pilote de stockage ontap-nas-flexgroup.
- Amélioration des performances du pilote de stockage économique ontap-nas lors de la gestion de plusieurs volumes FlexVol.
- Mises à jour des donnéesLIF activées pour tous les pilotes de stockage NAS de ONTAP.
- Mise à jour de la convention de nommage Trident Deployment and DemonSet afin de refléter le système d'exploitation du nœud hôte.

Dérations

- Kubernetes : mise à jour de Kubernetes minimale prise en charge vers la version 1.21.
- Les LIFs de données ne doivent plus être spécifiées lors de la configuration `ontap-san` ou `ontap-san-economy` pilotes.

Changements en 22.10

Vous devez lire les informations essentielles suivantes avant de passer à Astra Trident 22.10.

, la protection des données essentielles d'Astra Trident 22.10

- Kubernetes 1.25 est désormais pris en charge par Trident. Vous devez mettre à niveau Astra Trident vers 22.10 avant de procéder à la mise à niveau vers Kubernetes 1.25.
- Astra Trident applique désormais rigoureusement la configuration des chemins d'accès multiples dans les environnements SAN, avec la valeur recommandée de `find_multipaths: no` dans le fichier multipath.conf.

  Utilisation d'une configuration sans chemins d'accès multiples ou de l'utilisation de `find_multipaths: yes` ou `find_multipaths: smart` la valeur du fichier multipath.conf entraînera des échecs de montage. Trident a recommandé l'utilisation de `find_multipaths: no` depuis la version 21.07.

Correctifs

- Problème spécifique au système ONTAP back-end créé à l'aide de `credentials` le champ ne s'est pas connecté pendant la mise à niveau 22.07.0 ("Numéro 759").
- **Docker:** correction d'un problème entraînant l'échec du démarrage du plug-in de volume Docker dans certains environnements ("Numéro 548" et "Numéro 760").
- Résolution du problème SLM spécifique aux systèmes back-end ONTAP pour garantir que seul un sous-ensemble de LIF de données appartenant aux nœuds de reporting est publié.
- Problème de performances résolu lors de la connexion d'un volume à des analyses inutiles des LUN iSCSI.
- Supprimez les tentatives granulaires dans le workflow iSCSI Astra Trident pour connaître un risque d'échec rapide et réduire les intervalles de tentatives externes.
- Résolution du problème lorsqu'une erreur a été renvoyée lors du vidage d'un périphérique iSCSI lorsque le périphérique multivoie correspondant a déjà été rincé.

Améliorations

- Kubernetes :
  - Prise en charge ajoutée de Kubernetes 1.25. Vous devez mettre à niveau Astra Trident vers 22.10 avant de procéder à la mise à niveau vers Kubernetes 1.25.
  - Ajout d'un ServiceAccount, ClusterRole et ClusterRoleBinding distincts pour Trident Deployment et DemonSet afin de permettre des améliorations futures des autorisations.
  - Prise en charge ajoutée de "partage de volume entre espaces de noms".
- Tout Trident `ontap-*` Les pilotes de stockage fonctionnent désormais avec l'API REST de ONTAP.
- Ajout d'un nouvel opérateur yaml (`bundle_post_1_25.yaml`) sans a. `PodSecurityPolicy` Pour la prise en charge de Kubernetes 1.25.
- Ajouté "Prise en charge des volumes LUKS-chiffrés" pour `ontap-san` et `ontap-san-economy` lecteurs de stockage
- Ajout de la prise en charge des nœuds Windows Server 2019.
- Ajouté "Prise en charge des volumes SMB sur les nœuds Windows" grâce au `azure-netapp-files` pilote de stockage
- La détection automatique du basculement MetroCluster pour les pilotes ONTAP est désormais disponible dans l'ensemble.

Dérations

- **Kubernetes:** mise à jour du nombre minimum de Kubernetes pris en charge vers 1.20.
- Suppression du pilote ADS (Data Store).
- Retrait du support pour `yes` et `smart` options pour `find_multipaths` Lors de la configuration des chemins d'accès multiples du nœud de travail pour iSCSI.

Changements en 22.07

Correctifs

Kubernetes

- Problème résolu pour gérer les valeurs booléennes et nombres pour le sélecteur de nœud lors de la configuration de Trident avec Helm ou l'opérateur Trident. ("Problème GitHub n° 700")
- Résolution du problème lors de la gestion des erreurs provenant d'un chemin non CHAP, de sorte que kubelet réessaie en cas d'échec. "Problème GitHub n° 736")

Améliorations

- Passer de k8s.gcr.io au registre.k8s.io comme registre par défaut pour les images CSI
- Les volumes ONTAP-SAN utiliseront désormais des igroups par nœud et ne mapperont les LUN aux groupes initiateurs, tout en les ayant été publiés activement à ces nœuds pour améliorer notre sécurité. Lorsque Trident détermine que les volumes existants sont sécurisés, sans affecter les workloads actifs, les volumes existants seront transférés de manière opportuniste vers le nouveau modèle d'groupe initiateur.
- Inclus un quota de Resourcequota avec les installations Trident pour s'assurer que Trident DemonSet est planifié lorsque la consommation PriorityClass est limitée par défaut.

- Ajout de la prise en charge des fonctionnalités réseau au pilote ANF. ("Problème GitHub n° 717")
- Ajout de la détection automatique du basculement MetroCluster dans l'aperçu technique aux pilotes ONTAP. ("Problème GitHub n° 228")

Dérations

- **Kubernetes:** mise à jour du nombre minimum de Kubernetes pris en charge vers 1.19.
- La configuration backend n'autorise plus plusieurs types d'authentification dans la configuration unique.

Suppressions

- Le pilote CVS AWS (obsolète depuis 22.04) a été supprimé.
- Kubernetes
  - Suppression des capacités SYS_ADMIN inutiles des modules de nœud.
  - Réduit la préparation des nœuds afin de simplifier les informations sur l'hôte et la détection des services actifs pour obtenir la confirmation de la disponibilité des services NFS/iSCSI sur les nœuds workers.

Documentation

Une nouvelle "Normes de sécurité du pod" (PSS) a été ajouté en détail les autorisations activées par Astra Trident lors de l'installation.

Changements en 22.04

NetApp améliore et améliore continuellement ses produits et services. Voici quelques-unes des nouveautés d'Astra Trident. Pour les versions précédentes, voir "Versions antérieures de la documentation".

> Si vous effectuez une mise à niveau à partir d'une version précédente de Trident et que vous utilisez Azure NetApp Files, le `location` le paramètre config est désormais un champ singleton obligatoire.

Correctifs

- Amélioration de l'analyse des noms d'initiateurs iSCSI. ("Problème GitHub n° 681")
- Problème résolu lorsque les paramètres de classe de stockage CSI n'étaient pas autorisés. ("Problème GitHub n° 598")
- Déclaration de clé en double fixe dans Trident CRD. ("Problème GitHub n° 671")
- Correction des journaux CSI instantanés erronés. ("Problème GitHub n° 629"))
- Résolution du problème lié à l'annulation de la publication des volumes sur les nœuds supprimés. ("Problème GitHub n° 691")
- Ajout de la gestion des incohérences du système de fichiers sur les périphériques en bloc. ("Problème GitHub n° 656")
- Problème résolu extraction automatique des images lors de la configuration du `imageRegistry` indicateur pendant l'installation. ("Problème GitHub n° 715")
- Problème résolu où le pilote ANF n'a pas réussi à cloner un volume avec plusieurs règles d'exportation.

Améliorations

- Les connexions entrantes aux terminaux sécurisés de Trident requièrent désormais un minimum de TLS 1.3. ("Problème GitHub n° 698")
- Trident ajoute désormais des en-têtes HSTS aux réponses à partir de ses terminaux sécurisés.
- Trident tente désormais d'activer automatiquement la fonctionnalité d'autorisations unix Azure NetApp Files.
- **Kubernetes**: Trident demonset s'exécute maintenant dans la classe de priorité critique du nœud système. ("Problème GitHub n° 694")

Suppressions

Le pilote E-Series (désactivé depuis 20.07) a été supprimé.

Changements en 22.01.1

Correctifs

- Résolution du problème lié à l'annulation de la publication des volumes sur les nœuds supprimés. ("Problème GitHub n° 691")
- Panique fixe lors de l'accès aux champs nuls pour l'espace global dans les réponses de l'API ONTAP.

Changements en 22.01.0

Correctifs

- **Kubernetes:** augmentez le temps de rétentative de rétro-enregistrement des nœuds pour les grands clusters.
- Problème résolu dans lequel le pilote Azure-netapp-Files pourrait être confondu avec plusieurs ressources avec le même nom.
- Les LIF de données sur IPv6 SAN de ONTAP fonctionnent désormais si elles sont spécifiées avec des parenthèses.
- Problème résolu lors de la tentative d'importation d'un volume déjà importé renvoie EOF laissant le PVC à l'état en attente. ("Problème GitHub n° 489")
- Problème résolu lorsque la performance d'Astra Trident ralentit lorsque plus de 32 snapshots sont créés sur un volume SolidFire.
- SHA-1 remplacé par SHA-256 lors de la création du certificat SSL.
- Pilote ANF fixe pour autoriser les noms de ressource en double et limiter les opérations à un emplacement unique.
- Pilote ANF fixe pour autoriser les noms de ressource en double et limiter les opérations à un emplacement unique.

Améliorations

- Améliorations de Kubernetes :
  - Prise en charge ajoutée de Kubernetes 1.23.

- Ajoutez des options de planification pour les pods Trident lorsqu'ils sont installés via l'opérateur Trident ou Helm. ("Problème GitHub n° 651")

- Autorisation des volumes inter-régions dans le pilote GCP ("Problème GitHub n° 633")
- Ajout de la prise en charge de l'option 'unixpermissionss' aux volumes ANF. ("Problème GitHub n° 666")

Dérations

L'interface REST de Trident peut écouter et servir uniquement aux adresses 127.0.0.1 ou [::1]

Changements en 21.10.1

> La version v21.10.0 présente un problème qui peut placer le contrôleur Trident dans un état CrashLoopBackOff lorsqu'un nœud est supprimé, puis réintégré au cluster Kubernetes. Ce problème a été résolu dans la version 1.210.1 (édition GitHub 669).

Correctifs

- Condition de race potentielle fixe lors de l'importation d'un volume sur un back-end Cloud CVS GCP, entraînant l'échec de l'importation.
- Résolution d'un problème de mise en service du contrôleur Trident dans un état CashLoopBackOff lorsqu'un nœud est retiré, puis réintégré au cluster Kubernetes (problème GitHub 669).
- Problème résolu : les SVM n'ont plus été découverts si aucun nom de SVM n'a été spécifié (problème GitHub 612).

Changements en 21.10.0

Correctifs

- Problème résolu : les clones de volumes XFS n'ont pas pu être montés sur le même nœud que le volume source (problème GitHub 514).
- Résolution du problème pendant lequel Astra Trident a enregistré une erreur fatale lors de l'arrêt (problème GitHub 597).
- Correctifs liés à Kubernetes :
  - Renvoyer l'espace utilisé d'un volume comme taille de restauration minimale lors de la création de snapshots avec `ontap-nas` et `ontap-nas-flexgroup` Pilotes (problème GitHub 645).
  - Résolution du problème où `Failed to expand filesystem` Une erreur a été consignée après le redimensionnement du volume (problème GitHub 560).
  - Résolution du problème de blocage d'un module `Terminating` État (problème GitHub 572).
  - A résolu le cas où un `ontap-san-economy` FlexVol peut contenir des LUN de snapshot (GitHub : édition 533).
  - Résolution du problème d'installation YAML personnalisé avec une image différente (problème GitHub 613).
  - Calcul de la taille de snapshot fixe (problème GitHub 611).
  - Problème résolu : tous les installateurs Trident d'Astra pouvaient identifier Kubernetes ordinaire comme OpenShift (problème GitHub 639).

- A corrigé l'opérateur Trident pour arrêter la réconciliation si le serveur d'API Kubernetes est inaccessible (problème GitHub 599).

Améliorations

- Prise en charge ajoutée de `unixPermissions` Option pour les volumes de performance GCP-CVS.
- Ajout de la prise en charge des volumes CVS optimisés pour l'évolutivité dans GCP dans la plage de 600 Gio à 1 Tio.
- Améliorations liées à Kubernetes :
  - Prise en charge ajoutée de Kubernetes 1.22.
  - Compatibilité de l'opérateur Trident et du tableau Helm avec Kubernetes 1.22 (problème GitHub 628).
  - Ajout d'une image opérateur à `tridentctl` Commande images (problème GitHub 570).

Améliorations expérimentales

- Ajout de la prise en charge de la réplication de volume dans `ontap-san` conducteur.
- Ajout de la prise en charge de REST * TECH Preview* pour `ontap-nas-flexgroup`, `ontap-san`, et `ontap-nas-economy` pilotes.

Problèmes connus

Les problèmes connus identifient les problèmes susceptibles de vous empêcher d'utiliser le produit avec succès.

- Lorsque vous mettez à niveau un cluster Kubernetes de 1.24 vers 1.25 ou version ultérieure sur lequel Astra Trident est installé, vous devez mettre à jour les valeurs.yaml pour les définir `excludePodSecurityPolicy` à `true` ou ajouter `--set excludePodSecurityPolicy=true` à la `helm upgrade` commande avant de pouvoir mettre à niveau le cluster.
- L'ASTRA Trident applique maintenant une blanc `fsType` (`fsType=""`) pour les volumes qui n'ont pas le `fsType` Spécifiés dans leur classe de stockage. Lorsque vous utilisez Kubernetes 1.17 ou version ultérieure, Trident prend en charge l'option vide `fsType` Pour les volumes NFS. Pour les volumes iSCSI, vous devez définir le `fsType` Sur votre classe de stockage lors de l'application d'un `fsGroup` Utilisation d'un contexte de sécurité.
- Lors de l'utilisation d'un système back-end pour plusieurs instances Trident d'Astra, chaque fichier de configuration back-end doit avoir un fichier différent `storagePrefix` Valeur pour les systèmes ONTAP back-end ou différente `TenantName` Pour les systèmes SolidFire back-end. Astra Trident ne peut pas détecter les volumes que d'autres instances d'Astra Trident ont créés. Tentative de création d'un volume existant sur un système back-end ONTAP ou SolidFire réussie, Astra Trident traite la création de volume comme une opération identente. Si `storagePrefix` ou `TenantName` n'en diffère pas, il peut y avoir des collisions de noms pour les volumes créés sur le même back-end.
- Lors de l'installation d'Astra Trident (à l'aide de `tridentctl` Ou l'opérateur Trident) et à l'aide de `tridentctl` Pour gérer Astra Trident, vous devez vous assurer que `KUBECONFIG` la variable d'environnement est définie. Cela est nécessaire pour indiquer le cluster Kubernetes `tridentctl` doit travailler contre. Lorsque vous utilisez plusieurs environnements Kubernetes, assurez-vous que `KUBECONFIG` le fichier est fourni avec précision.
- Pour réclamer de l'espace en ligne pour des volumes persistants iSCSI, le système d'exploitation sous-jacent du nœud worker peut nécessiter le passage des options de montage vers le volume. Ceci est vrai

pour les instances RHEL/RedHat CoreOS qui requièrent le `discard` "option de montage"; Assurez-vous que le mountOption de mise au rebut est inclus dans votre[`StorageClass`^] pour prendre en charge le blocage en ligne.

- Si vous possédez plusieurs instances d'Astra Trident par cluster Kubernetes, Astra Trident ne peut pas communiquer avec d'autres instances et ne peut pas détecter les autres volumes qu'ils ont créés, ce qui entraîne un comportement inattendu et incorrect si plusieurs instances s'exécutent dans un cluster. Il ne devrait y avoir qu'une seule instance d'Astra Trident par cluster Kubernetes.
- Avec Astra Trident `StorageClass` Les objets sont supprimés de Kubernetes alors que Astra Trident est hors ligne, Astra Trident ne supprime pas les classes de stockage correspondantes de la base de données lorsqu'elle est remise en ligne. Vous devez supprimer ces classes de stockage à l'aide de `tridentctl` Ou l'API REST.
- Si un utilisateur supprime un volume persistant provisionné par Astra Trident avant de supprimer le volume persistant correspondant, Astra Trident ne supprime pas automatiquement le volume de sauvegarde. Vous devez supprimer le volume via `tridentctl` Ou l'API REST.
- ONTAP ne peut pas provisionner simultanément plusieurs FlexGroup, sauf si l'ensemble d'agrégats est unique pour chaque demande de provisionnement.
- Lorsque vous utilisez Astra Trident sur IPv6, vous devez préciser `managementLIF` et `dataLIF` dans la définition du back-end entre crochets. Par exemple : `[fd20:8b1e:b258:2000:f816:3eff:feec:0]`.

> Vous ne pouvez pas spécifier `dataLIF` Sur un SAN backend ONTAP. Astra Trident détecte toutes les LIF iSCSI disponibles et les utilise pour établir la session multivoie.

- Si vous utilisez le `solidfire-san` Pilote avec OpenShift 4.5, assurez-vous que les nœuds de travail sous-jacents utilisent MD5 comme algorithme d'authentification CHAP. Les algorithmes CHAP sécurisés conformes à la norme FIPS SHA1, SHA-256 et SHA3-256 sont disponibles avec Element 12.7.

Trouvez plus d'informations

- "Astra Trident GitHub"
- "Blogs Trident d'Astra"

Concepts

Découvrez Astra Trident

Astra Trident est un projet open source entièrement pris en charge, géré par NetApp dans le cadre du "Gamme de produits Astra". Il a été conçu pour vous aider à répondre aux demandes de persistance de vos applications conteneurisées à l'aide d'interfaces standard, telles que l'interface de stockage de conteneurs (CSI).

Présentation

Astra Trident déploie dans les clusters Kubernetes en tant que pods et fournit des services d'orchestration du stockage dynamique pour vos workloads Kubernetes. Il permet à vos applications conteneurisées de consommer rapidement et facilement le stockage persistant à partir du vaste portefeuille de NetApp, notamment ONTAP (AFF/FAS/Select/Cloud/Amazon FSX pour NetApp ONTAP), le logiciel Element (NetApp HCI/SolidFire), le service Azure NetApp Files et Cloud Volumes Service sur Google Cloud.

Astra Trident est également une technologie fondamentale de NetApp Astra, qui répond à vos besoins en matière de protection des données, de reprise après incident, de portabilité et de migration pour les workloads Kubernetes, en exploitant la technologie leader de gestion des données NetApp pour les copies Snapshot, la sauvegarde, la réplication et le clonage.

Architectures de cluster Kubernetes prises en charge

Astra Trident est pris en charge avec les architectures Kubernetes suivantes :

Architectures en cluster Kubernetes	Pris en charge	Installation par défaut
Maître unique, calcul	Oui.	Oui.
Plusieurs maîtres, calcul	Oui.	Oui.
Maître, `etcd`, calculer	Oui.	Oui.
Maîtrise, infrastructure, calcul	Oui.	Oui.

Qu'est-ce qu'Astra ?

Astra facilite la gestion, la protection et le déplacement de leurs workloads riches en données qui s'exécutent sur Kubernetes, dans les clouds publics et sur site. Astra provisionne et fournit un stockage persistant pour les conteneurs utilisant Astra Trident, un portefeuille de stockage étendu et éprouvé 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.

Vous pouvez vous inscrire pour un essai gratuit sur la page Astra.

Pour en savoir plus

- "Gamme NetApp Astra"
- "Documentation relative au service après-vente Astra Control"
- "Documentation Astra Control Center"
- "Documentation de l'API Astra"

Pilotes ONTAP

Astra Trident propose cinq pilotes de stockage ONTAP uniques pour la communication avec les clusters ONTAP. Découvrez comment chaque pilote gère la création de volumes, le contrôle d'accès et leurs fonctionnalités.

Découvrez les pilotes de stockage ONTAP

Astra Control assure une protection, une reprise d'activité et une mobilité transparentes (en déplaçant des volumes entre les clusters Kubernetes) pour les volumes créés avec le système `ontap-nas`, `ontap-nas-flexgroup`, et `ontap-san` pilotes. Voir "Conditions préalables à la réplication d'Astra Control" pour plus d'informations.

- Vous devez utiliser `ontap-nas` adapté aux charges de travail de production qui nécessitent une protection des données, une reprise d'activité et la mobilité.
- Utiliser `ontap-san-economy` Lorsque vous prévoyez une utilisation de volume, celle-ci devrait être bien supérieure à celle prise en charge par ONTAP.
- Utiliser `ontap-nas-economy` Ce n'est que lorsque l'utilisation prévue des volumes sera beaucoup plus élevée que ce que prend en charge ONTAP, et le `ontap-san-economy` le pilote ne peut pas être utilisé.
- Ne pas utiliser `ontap-nas-economy` si vous prévoyez d'avoir besoin en termes de protection des données, de reprise sur incident ou de mobilité.

Conducteur	Protocole	Mode Volume	Modes d'accès pris en charge	Systèmes de fichiers pris en charge
`ontap-nas`	NFS	Système de fichiers	RWO,ROX,RWX	« », nfs
`ontap-nas-economy`	NFS	Système de fichiers	RWO,ROX,RWX	« », nfs
`ontap-nas-flexgroup`	NFS	Système de fichiers	RWO,ROX,RWX	« », nfs
`ontap-san`	ISCSI	Bloc	RWO,ROX,RWX	Pas de système de fichiers, périphérique de bloc brut

Conducteur	Protocole	Mode Volume	Modes d'accès pris en charge	Systèmes de fichiers pris en charge
`ontap-san`	ISCSI	Système de fichiers	RWO,ROX RWX n'est pas disponible en mode de volume du système de fichiers.	`xfs`, `ext3`, `ext4`
`ontap-san-economy`	ISCSI	Bloc	RWO,ROX,RWX	Pas de système de fichiers, périphérique de bloc brut
`ontap-san-economy`	ISCSI	Système de fichiers	RWO,ROX RWX n'est pas disponible en mode de volume du système de fichiers.	`xfs`, `ext3`, `ext4`

Les systèmes back-end ONTAP peuvent être authentifiés avec des identifiants de connexion pour un rôle de sécurité (nom d'utilisateur/mot de passe) ou à l'aide de la clé privée et du certificat installés sur le cluster ONTAP. Vous pouvez mettre à jour les systèmes back-end existants pour passer d'un mode d'authentification à l'autre avec `tridentctl update backend`.

Provisionnement

Le provisionnement avec Astra Trident a deux phases principales. La première phase associe une classe de stockage à l'ensemble des pools de stockage back-end appropriés et effectue la préparation nécessaire avant le provisionnement. La deuxième phase inclut la création du volume lui-même et nécessite le choix d'un pool de stockage à partir de ceux associés à la classe de stockage du volume en attente.

Association de classe de stockage

L'association de pools de stockage back-end à une classe de stockage repose sur les attributs demandés par la classe de stockage et sur leur `storagePools`, `additionalStoragePools`, et `excludeStoragePools` listes. Lorsque vous créez une classe de stockage, Trident compare les attributs et les pools proposés par chacun de ses systèmes back-end à ceux requis par la classe de stockage. Si les attributs et le nom d'un pool de stockage correspondent à tous les attributs et noms de pool demandés, Astra Trident ajoute ce pool de stockage à l'ensemble des pools de stockage appropriés pour cette classe de stockage. D'autre part, Astra Trident ajoute tous les pools de stockage répertoriés dans le `additionalStoragePools` énumérez cet ensemble, même si leurs attributs ne remplissent pas tous ou aucun des attributs demandés par la classe de stockage. Vous devez utiliser le `excludeStoragePools` liste pour remplacer et supprimer les pools de stockage utilisés pour une classe de stockage. La solution Astra Trident effectue un processus similaire chaque fois que vous ajoutez un nouveau système back-end, en vérifiant si ses pools de stockage

correspondent à ceux des classes de stockage existantes et en supprimant ceux qui ont été marqués comme exclus.

Création du volume

Astra Trident utilise ensuite les associations entre les classes de stockage et les pools de stockage pour déterminer où provisionner les volumes. Lorsque vous créez un volume, Astra Trident commence par obtenir l'ensemble des pools de stockage correspondant à la classe de stockage du volume. De plus, si vous spécifiez un protocole pour le volume, Astra Trident supprime les pools de stockage qui ne peuvent pas fournir le protocole demandé (par exemple, un back-end NetApp HCI/SolidFire ne peut pas fournir un volume basé sur les fichiers, tandis qu'un back-end ONTAP NAS ne peut pas fournir un volume basé sur les blocs). Astra Trident répartit de manière aléatoire l'ordre de ce jeu, afin de faciliter une distribution homogène des volumes, puis l'itérate via lui pour tenter de provisionner le volume sur chaque pool de stockage. S'il réussit sur un, il retourne avec succès, en enregistrant les échecs rencontrés dans le processus. Astra Trident renvoie une défaillance **uniquement si** il ne parvient pas à approvisionner **tous** les pools de stockage disponibles pour la classe et le protocole de stockage demandés.

Snapshots de volume

Découvrez comment Astra Trident gère la création de copies Snapshot de volume pour ses pilotes.

En savoir plus sur la création de snapshots de volume

- Pour le `ontap-nas`, `ontap-san`, `gcp-cvs`, et `azure-netapp-files` Chaque volume persistant est mappé à un FlexVol. Par conséquent, des snapshots de volume sont créés sous forme de snapshots NetApp. La technologie Snapshot de NetApp offre une stabilité, une évolutivité, une capacité de restauration et des performances supérieures à celles des technologies snapshot concurrentes. Ces copies Snapshot sont extrêmement efficaces, aussi bien en termes de temps de création que d'espace de stockage.
- Pour le `ontap-nas-flexgroup` Chaque pilote de volume persistant est mappé à un FlexGroup. Par conséquent, des snapshots de volume sont créés sous forme de snapshots NetApp FlexGroup. La technologie Snapshot de NetApp offre une stabilité, une évolutivité, une capacité de restauration et des performances supérieures à celles des technologies snapshot concurrentes. Ces copies Snapshot sont extrêmement efficaces, aussi bien en termes de temps de création que d'espace de stockage.
- Pour le `ontap-san-economy` Le pilote et les volumes persistants sont mis en correspondance avec les LUN créées sur les volumes FlexVol partagés. Les copies FlexClone de la LUN associée permettent d'obtenir les copies Snapshot VolumeSnapshot de la LUN associée. La technologie FlexClone d'ONTAP permet de créer quasi instantanément des copies de jeux de données, même les plus volumineux. Les copies partagent les blocs de données avec leurs parents. Aucun stockage n'est nécessaire, sauf pour les métadonnées.
- Pour le `solidfire-san` Pilote, chaque volume persistant est mappé sur une LUN créée dans le cluster NetApp Element/NetApp HCI. Les copies Snapshot VolumeCas sont représentées par des copies Snapshot Element de la LUN sous-jacente. Ces snapshots sont des copies à un point dans le temps et ne prennent en charge qu'une petite quantité de ressources et d'espace système.
- Lorsque vous travaillez avec le `ontap-nas` et `ontap-san` Les snapshots ONTAP sont des copies ponctuelles de la FlexVol et consomment de l'espace sur la FlexVol elle-même. Cela peut entraîner la quantité d'espace inscriptible dans le volume pour une réduction du temps lors de la création ou de la planification des snapshots. L'une des façons simples de résoudre ce problème est d'augmenter le volume en le redimensionnant via Kubernetes. Une autre option consiste à supprimer les snapshots qui ne sont plus nécessaires. Lors de la suppression d'un Snapshot VolumeCas créé via Kubernetes, Astra Trident

supprime le snapshot ONTAP associé. Les snapshots ONTAP qui n'ont pas été créés par Kubernetes peuvent également être supprimés.

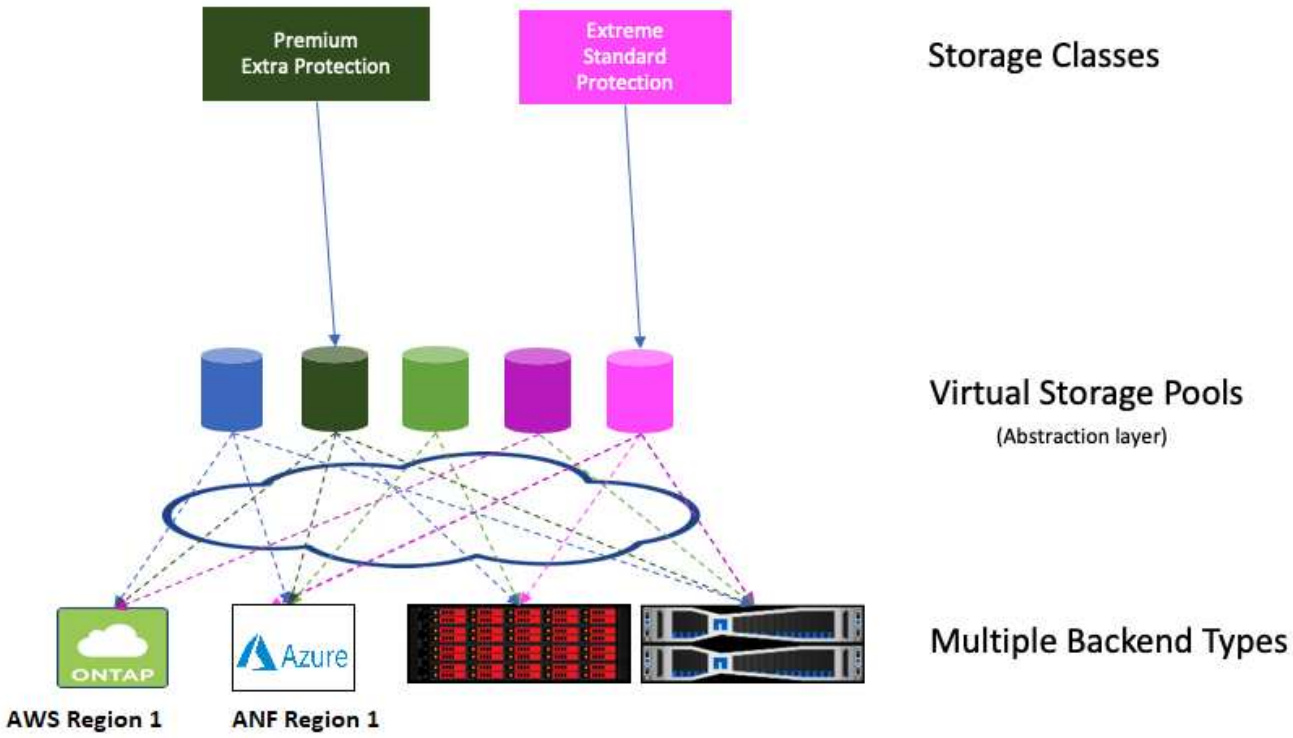
Avec Astra Trident, vous pouvez utiliser Volumesnapshots pour créer de nouveaux volumes persistants à partir d'entre eux. La création de volumes persistants est effectuée à partir de ces copies Snapshot à l'aide de la technologie FlexClone pour les systèmes back-end ONTAP et CVS pris en charge. Lors de la création d'un volume persistant à partir d'un snapshot, le volume de sauvegarde est un volume FlexClone du volume parent du snapshot. Le `solidfire-san` Le pilote utilise des clones de volumes logiciels Element pour créer des volumes persistants à partir de snapshots. Ici, cela crée un clone à partir du snapshot Element.

Pools virtuels

Les pools virtuels fournissent une couche d'abstraction entre les systèmes back-end de stockage Astra Trident et Kubernetes `StorageClasses`. Ils permettent à un administrateur de définir des aspects, tels que l'emplacement, les performances et la protection pour chaque système back-end, de façon commune et indépendante du système back-end `StorageClass` spécifiez le type de back-end physique, de pool back-end ou de type back-end à utiliser pour répondre aux critères souhaités.

En savoir plus sur les pools virtuels

L'administrateur du stockage peut définir des pools virtuels sur n'importe quel système back-end Trident Astra dans un fichier de définition JSON ou YAML.



Tout aspect spécifié en dehors de la liste des pools virtuels est global au back-end et s'appliquera à tous les pools virtuels, tandis que chaque pool virtuel peut spécifier un ou plusieurs aspects individuellement (remplaçant les aspects backend-global).

- Lors de la définition de pools virtuels, n'essayez pas de réorganiser l'ordre des pools virtuels existants dans une définition backend.
- Nous vous conseillons de modifier les attributs d'un pool virtuel existant. Vous devez définir un nouveau pool virtuel pour apporter des modifications.

La plupart des aspects sont spécifiés dans des termes spécifiques au système back-end. Il est primordial que les valeurs de l'aspect ne soient pas exposées en dehors du conducteur du back-end et ne soient pas disponibles pour la correspondance dans `StorageClasses`. À la place, l'administrateur définit un ou plusieurs libellés pour chaque pool virtuel. Chaque étiquette est une paire clé:valeur et les étiquettes sont souvent répandues sur différents systèmes back-end. Tout comme les aspects, les étiquettes peuvent être spécifiées par pool ou globales au back-end. Contrairement aux aspects, qui ont des noms et des valeurs prédéfinis, l'administrateur dispose d'une entière discrétion pour définir les clés et les valeurs de libellé selon les besoins. Pour plus de commodité, les administrateurs du stockage peuvent définir des étiquettes par pool virtuel et les volumes de groupe par étiquette.

A `StorageClass` identifie le pool virtuel à utiliser en référençant les étiquettes dans un paramètre de sélection. Les sélecteurs de pool virtuel prennent en charge les opérateurs suivants :

Opérateur	Exemple	La valeur d'étiquette d'un pool doit :
`=`	performance=premium	Correspondance
`!=`	performance !=extrême	Ne correspond pas
`in`	emplacement à (est, ouest)	Être dans l'ensemble de valeurs
`notin`	performances notin (argent, bronze)	Ne pas être dans l'ensemble de valeurs
`<key>`	la protection	Existe avec n'importe quelle valeur
`!<key>`	!protection	N'existe pas

Groupes d'accès de volume

Découvrez l'utilisation d'Astra Trident "groupes d'accès de volume".

Ignorez cette section si vous utilisez CHAP, qui est recommandé pour simplifier la gestion et éviter la limite de mise à l'échelle décrite ci-dessous. De plus, si vous utilisez Astra Trident en mode CSI, vous pouvez ignorer cette section. Astra Trident utilise CHAP lorsqu'il est installé en tant que mécanisme de provisionnement CSI amélioré.

En savoir plus sur les groupes d'accès aux volumes

Astra Trident peut utiliser des groupes d'accès de volume pour contrôler l'accès aux volumes qu'il provisionne. Si CHAP est désactivé, il attend de trouver un groupe d'accès appelé `trident` Sauf si vous spécifiez un ou plusieurs ID de groupe d'accès dans la configuration.

Astra Trident associe de nouveaux volumes aux groupes d'accès configurés, mais il ne crée ni ne gère d'autre manière les groupes d'accès eux-mêmes. Le ou les groupes d'accès doivent exister avant l'ajout du système de stockage back-end à Astra Trident. Ils doivent également contenir les IQN iSCSI de chaque nœud du cluster Kubernetes qui pourraient monter les volumes provisionnés par ce système back-end. Dans la plupart des installations, cela inclut tous les nœuds workers dans le cluster.

Pour les clusters Kubernetes de plus de 64 nœuds, vous devez utiliser plusieurs groupes d'accès. Chaque groupe d'accès peut contenir jusqu'à 64 IQN et chaque volume peut appartenir à quatre groupes d'accès. Avec quatre groupes d'accès configurés au maximum, n'importe quel nœud d'un cluster de 256 nœuds maximum pourra accéder à n'importe quel volume. Pour connaître les dernières limites des groupes d'accès aux volumes, reportez-vous à la section "ici".

Si vous modifiez la configuration à partir d'une configuration qui utilise la valeur par défaut `trident` Groupe d'accès à un groupe qui utilise également d'autres, inclure l'ID pour le `trident` groupe d'accès dans la liste.

Commencez

Essayez-le

NetApp vous offre une image du laboratoire prêt à l'emploi que vous pouvez demander via "Test Drive NetApp".

Découvrez Test Drive

Test Drive vous fournit un environnement sandbox fourni avec un cluster Kubernetes à trois nœuds et Astra Trident installé et configuré. C'est un excellent moyen de vous familiariser avec Astra Trident et d'explorer ses caractéristiques.

Une autre option est de voir "Guide d'installation de kubeadm" Fourni par Kubernetes.

> Vous ne devez pas utiliser le cluster Kubernetes que vous créez à l'aide de ces instructions en production. Utilisez les guides de déploiement de production fournis par votre distribution pour créer des clusters prêts à l'emploi.

Si c'est la première fois que vous utilisez Kubernetes, familiarisez-vous avec les concepts et les outils "ici".

De formation

Avant d'installer Astra Trident, il est recommandé de vérifier ces exigences système générales. Il se peut que les systèmes back-end spécifiques présentent des exigences supplémentaires.

Découvrez Astra Trident 23.01, un document essentiel

Vous devez lire les renseignements essentiels suivants sur Astra Trident.

** informations pratiques sur le Tridécouvrez Astra **

- Kubernetes 1.26 est désormais pris en charge par Trident. Mise à niveau de Trident avant la mise à niveau de Kubernetes.
- Astra Trident applique rigoureusement l'utilisation de la configuration de chemins d'accès multiples dans les environnements SAN, avec la valeur recommandée de `find_multipaths: no` dans le fichier multipath.conf.

  Utilisation d'une configuration sans chemins d'accès multiples ou de l'utilisation de `find_multipaths: yes` ou `find_multipaths: smart` la valeur du fichier multipath.conf entraînera des échecs de montage. Trident a recommandé l'utilisation de `find_multipaths: no` depuis la version 21.07.

Systèmes front-end (orchestrateurs) pris en charge

Astra Trident prend en charge plusieurs moteurs et orchestrateurs de conteneur, notamment :

- Anthos sur site (VMware) et Anthos sur un système bare Metal 1.9, 1.10, 1.11
- Kubernetes 1.21 - 1.26
- Mirantis Kubernetes Engine 3.5
- OpenShift 4.9 - 4.12

L'opérateur de Trident est pris en charge par ces versions :

- Anthos sur site (VMware) et Anthos sur un système bare Metal 1.9, 1.10, 1.11
- Kubernetes 1.21 - 1.26
- OpenShift 4.9 - 4.12

Astra Trident fonctionne également avec d'autres offres Kubernetes autogérées et entièrement gérées, notamment Google Kubernetes Engine (GKE), Amazon Elastic Kubernetes Services (EKS), Azure Kubernetes Service (AKS), Rancher et VMware Tanzu Portfolio.

Avant de mettre à niveau un cluster Kubernetes de la version 1.24 à la version 1.25 ou ultérieure avec Astra Trident installé, consultez la page "Mettre à niveau l'installation d'un opérateur basé sur Helm".

Systèmes back-end pris en charge (stockage)

Pour utiliser Astra Trident, vous avez besoin d'un ou de plusieurs des systèmes back-end pris en charge :

- Amazon FSX pour NetApp ONTAP
- Azure NetApp Files
- Cloud Volumes ONTAP
- Cloud Volumes Service pour GCP
- FAS/AFF/Select 9.5 ou version ultérieure
- Baie SAN 100 % Flash (ASA) de NetApp
- Logiciel NetApp HCI/Element 11 ou version ultérieure

Configuration requise

Le tableau ci-dessous résume les fonctionnalités disponibles dans cette version d'Astra Trident et les versions de Kubernetes qu'il prend en charge.

Fonction	Version Kubernetes	Portes-fonctions requises ?
CSI Trident	1.21 - 1.26	Non
Snapshots de volume	1.21 - 1.26	Non
Volume persistant à partir des copies Snapshot des volumes	1.21 - 1.26	Non
Redimensionnement PV iSCSI	1.21 - 1.26	Non

Fonction	Version Kubernetes	Portes-fonctions requises ?
Chap bidirectionnel ONTAP	1.21 - 1.26	Non
Règles d'exportation dynamiques	1.21 - 1.26	Non
Opérateur Trident	1.21 - 1.26	Non
Topologie CSI	1.21 - 1.26	Non

Systèmes d'exploitation hôtes testés

Bien que l'Astra Trident ne prenne pas officiellement en charge des systèmes d'exploitation spécifiques, il s'agit des éléments suivants qui fonctionnent :

- Versions Red Hat CoreOS (RHCOS) prises en charge par OpenShift Container Platform
- RHEL 8+
- Ubuntu 22.04 ou version ultérieure
- Windows Server 2019

Par défaut, Astra Trident s'exécute dans un conteneur et s'exécute donc sur un utilisateur Linux. Cependant, ces employés doivent pouvoir monter les volumes qu'Astra Trident utilise le client NFS standard ou l'initiateur iSCSI, en fonction du système back-end utilisé.

Le `tridentctl` Utility s'exécute également sur l'une de ces distributions de Linux.

Configuration de l'hôte

Tous les nœuds workers du cluster Kubernetes doivent pouvoir monter les volumes provisionnés pour vos pods. Pour préparer les nœuds workers, vous devez installer des outils NFS ou iSCSI en fonction de votre sélection de pilotes.

"Préparez le nœud de travail"

Configuration du système de stockage

Il est possible qu'Astra Trident modifie le système de stockage avant qu'une configuration back-end ne puisse l'utiliser.

"Configuration des systèmes back-end"

Ports Trident d'Astra

L'Astra Trident doit accéder à des ports spécifiques pour la communication.

"Ports Trident d'Astra"

Images de conteneur et versions Kubernetes correspondantes

Pour les installations utilisant des systèmes à air comprimé, la liste suivante est une référence des images de conteneur nécessaires à l'installation d'Astra Trident. Utilisez le `tridentctl images` commande pour vérifier la liste des images de conteneur requises.

Version Kubernetes	Image de conteneur
v1.21.0	• docker.io/netapp/trident : 23.01.1 • docker.io/netapp/trident-autosupport:23.01 • registry.k8s.io/sig-storage/csi-provisionneur:v3.4.0 • registry.k8s.io/sig-storage/csi-attacher:v4.1.0 • registry.k8s.io/sig-storage/csi-resizer:v1.7.0 • registry.k8s.io/sig-storage/csi-snapshotter:v6.2.1 • registry.k8s.io/sig-storage/csi-node-driver-registratr:v2.7.0 • docker.io/netapp/trident-operator:23.01.1 (en option)
v1.22.0	• docker.io/netapp/trident : 23.01.1 • docker.io/netapp/trident-autosupport:23.01 • registry.k8s.io/sig-storage/csi-provisionneur:v3.4.0 • registry.k8s.io/sig-storage/csi-attacher:v4.1.0 • registry.k8s.io/sig-storage/csi-resizer:v1.7.0 • registry.k8s.io/sig-storage/csi-snapshotter:v6.2.1 • registry.k8s.io/sig-storage/csi-node-driver-registratr:v2.7.0 • docker.io/netapp/trident-operator:23.01.1 (en option)
v1.23.0	• docker.io/netapp/trident : 23.01.1 • docker.io/netapp/trident-autosupport:23.01 • registry.k8s.io/sig-storage/csi-provisionneur:v3.4.0 • registry.k8s.io/sig-storage/csi-attacher:v4.1.0 • registry.k8s.io/sig-storage/csi-resizer:v1.7.0 • registry.k8s.io/sig-storage/csi-snapshotter:v6.2.1 • registry.k8s.io/sig-storage/csi-node-driver-registratr:v2.7.0 • docker.io/netapp/trident-operator:23.01.1 (en option)

Version Kubernetes	Image de conteneur
v1.24.0	• docker.io/netapp/trident : 23.01.1 • docker.io/netapp/trident-autosupport:23.01 • registry.k8s.io/sig-storage/csi-provisionneur:v3.4.0 • registry.k8s.io/sig-storage/csi-attacher:v4.1.0 • registry.k8s.io/sig-storage/csi-resizer:v1.7.0 • registry.k8s.io/sig-storage/csi-snapshotter:v6.2.1 • registry.k8s.io/sig-storage/csi-node-driver-registratr:v2.7.0 • docker.io/netapp/trident-operator:23.01.1 (en option)
v1.25.0	• docker.io/netapp/trident : 23.01.1 • docker.io/netapp/trident-autosupport:23.01 • registry.k8s.io/sig-storage/csi-provisionneur:v3.4.0 • registry.k8s.io/sig-storage/csi-attacher:v4.1.0 • registry.k8s.io/sig-storage/csi-resizer:v1.7.0 • registry.k8s.io/sig-storage/csi-snapshotter:v6.2.1 • registry.k8s.io/sig-storage/csi-node-driver-registratr:v2.7.0 • docker.io/netapp/trident-operator:23.01.1 (en option)
v1.26.0	• docker.io/netapp/trident : 23.01.1 • docker.io/netapp/trident-autosupport:23.01 • registry.k8s.io/sig-storage/csi-provisionneur:v3.4.0 • registry.k8s.io/sig-storage/csi-attacher:v4.1.0 • registry.k8s.io/sig-storage/csi-resizer:v1.7.0 • registry.k8s.io/sig-storage/csi-snapshotter:v6.2.1 • registry.k8s.io/sig-storage/csi-node-driver-registratr:v2.7.0 • docker.io/netapp/trident-operator:23.01.1 (en option)

Sur Kubernetes version 1.21 et supérieure, utilisez la solution validée `registry.k8s.gcr.io/sig-storage/csi-snapshotter:v6.x` image uniquement si `v1` la version sert le `volumesnapshots.snapshot.storage.k8s.gcr.io` CRD. Si le `v1beta1` La version sert le CRD avec/sans le `v1` utilisez la version validée `registry.k8s.gcr.io/sig-storage/csi-snapshotter:v3.x` image.

Installer Astra Trident

Découvrez l'installation d'Astra Trident

Pour vérifier qu'Astra Trident peut être installé dans un grand nombre d'environnements et d'entreprises, NetApp propose plusieurs options d'installation. Vous pouvez installer Astra Trident à l'aide de l'opérateur Trident (manuellement ou à l'aide de Helm) ou à l'aide de `tridentctl`. Cette rubrique fournit des informations importantes pour sélectionner le processus d'installation qui vous convient.

Découvrez Astra Trident 23.01, un document essentiel

Vous devez lire les renseignements essentiels suivants sur Astra Trident.

** informations pratiques sur le Tridécouvrez Astra **

- Kubernetes 1.26 est désormais pris en charge par Trident. Mise à niveau de Trident avant la mise à niveau de Kubernetes.
- Astra Trident applique rigoureusement l'utilisation de la configuration de chemins d'accès multiples dans les environnements SAN, avec la valeur recommandée de `find_multipaths: no` dans le fichier multipath.conf.

  Utilisation d'une configuration sans chemins d'accès multiples ou de l'utilisation de `find_multipaths: yes` ou `find_multipaths: smart` la valeur du fichier multipath.conf entraînera des échecs de montage. Trident a recommandé l'utilisation de `find_multipaths: no` depuis la version 21.07.

Avant de commencer

Quel que soit votre chemin d'installation, vous devez avoir :

- Privilèges complets vers un cluster Kubernetes pris en charge exécutant une version prise en charge de Kubernetes et conditions requises pour les fonctionnalités activées. Vérifiez le "de formation" pour plus d'informations.
- Accès à un système de stockage NetApp pris en charge.
- Capacité de monter des volumes à partir de tous les nœuds de travail Kubernetes.
- Un hôte Linux avec `kubectl` (ou `oc`, Si vous utilisez OpenShift) installé et configuré pour gérer le cluster Kubernetes que vous souhaitez utiliser.
- Le `KUBECONFIG` Variable d'environnement qui pointe vers votre configuration de cluster Kubernetes.
- Si vous utilisez Kubernetes avec Docker Enterprise, "Suivez les étapes indiquées pour activer l'accès à l'interface de ligne de commande".

Si vous ne vous êtes pas familiarisé avec le "concepts de base", c'est le moment idéal pour le faire.

Choisissez votre méthode d'installation

Sélectionnez la méthode d'installation qui vous convient. Vous devez également examiner les considérations à prendre en compte pour "passage d'une méthode à l'autre" avant de prendre votre décision.

Utilisation de l'opérateur Trident

Que ce soit pour un déploiement manuel ou à l'aide de Helm, l'opérateur Trident est un excellent moyen de simplifier l'installation et de gérer dynamiquement les ressources Trident. Vous pouvez même "Personnalisez le déploiement de l'opérateur Trident" utilisation des attributs dans `TridentOrchestrator` Ressource personnalisée (CR).

L'utilisateur de Trident présente les avantages suivants :

** de l'objet de la carte de Trident crece **

L'opérateur Trident crée automatiquement les objets suivants pour votre version Kubernetes.

- ServiceAccount pour l'opérateur
- ClusterRole et ClusterRoleBinding au ServiceAccount
- Dedicated PodSecurityPolicy (pour Kubernetes 1.25 et versions antérieures)
- L'opérateur lui-même

** - Capcuratif de la prise **

L'opérateur surveille l'installation d'Astra Trident et prend activement des mesures pour résoudre les problèmes, par exemple lorsque le déploiement est supprimé ou lorsqu'il est modifié par erreur. A `trident-operator-<generated-id>` le pod est créé et associe un `TridentOrchestrator` CR avec une installation Astra Trident. Cela garantit qu'il n'y a qu'une seule instance d'Astra Trident dans le cluster et contrôle sa configuration, en s'assurant que l'installation est idemopuissante. Lorsque des modifications sont apportées à l'installation (par exemple, la suppression du déploiement ou du demonset de nœuds), l'opérateur les identifie et les corrige individuellement.

** mise à jour de l'installation de existante**

Vous pouvez facilement mettre à jour un déploiement existant avec l'opérateur. Il vous suffit de modifier le `TridentOrchestrator` CR pour effectuer des mises à jour d'une installation.

Prenons l'exemple d'un scénario dans lequel vous devez activer Astra Trident pour générer des journaux de débogage. Pour ce faire, patch de votre `TridentOrchestrator` à régler `spec.debug` à `true`:

```
kubectl patch torc <trident-orchestrator-name> -n trident --type=merge
-p '{"spec":{"debug":true}}'
```

Après `TridentOrchestrator` est mis à jour, l'opérateur traite les mises à jour et met à jour l'installation existante. Cela peut déclencher la création de nouveaux modules pour modifier l'installation en conséquence.

** : mise à niveau de **

Lorsque la version Kubernetes du cluster est mise à niveau vers une version prise en charge, l'opérateur met automatiquement à jour une installation Astra Trident existante et la modifie pour s'assurer qu'elle répond aux exigences de la version Kubernetes.

> Si le cluster est mis à niveau vers une version non prise en charge, l'opérateur empêche l'installation d'Astra Trident. Si Astra Trident a déjà été installé avec l'opérateur, un avertissement s'affiche pour indiquer que l'Astra Trident est installé sur une version Kubernetes non prise en charge.

Gestion de clusters ™ avec BlueXP (anciennement Cloud Manager)

Avec "Astra Trident avec BlueXP", Vous pouvez effectuer la mise à niveau vers la dernière version d'Astra Trident, ajouter et gérer des classes de stockage, les connecter aux environnements de travail et sauvegarder des volumes persistants à l'aide de Cloud Backup Service. BlueXP prend en charge le déploiement Astra Trident à l'aide de l'opérateur Trident, soit manuellement, soit via Helm.

À l'aide de `tridentctl`

Si votre déploiement existant doit être mis à niveau ou si vous cherchez à le personnaliser hautement, il est à votre disposition . Il s'agit de la méthode classique de déploiement d'Astra Trident.

C'est possible Générer les manifestes pour les ressources Trident. Cela inclut le déploiement, la demonset, le compte de service et le rôle de cluster qu'Astra Trident a créé dans le cadre de son installation.

> Depuis la version 22.04, les clés AES ne sont plus régénérées à chaque installation d'Astra Trident. Avec cette version, Astra Trident va installer un nouvel objet secret qui persiste dans toutes les installations. Cela signifie que `tridentctl` La version 22.04 peut désinstaller les versions précédentes de Trident, mais les versions antérieures ne peuvent pas désinstaller 22.04 installations. Sélectionnez l'installation appropriée *method*.

Choisissez votre mode d'installation

Déterminez votre processus de déploiement en fonction du *mode d'installation* (Standard, Offline ou Remote) requis par votre organisation.

Installation standard

Il s'agit du moyen le plus simple d'installer Astra Trident et qui fonctionne pour la plupart des environnements qui n'imposent pas de restrictions de réseau. Le mode d'installation standard utilise les registres par défaut pour stocker Trident requis (`docker.io`) Et CSI (`registry.k8s.io`) images.

Lorsque vous utilisez le mode standard, le programme d'installation d'Astra Trident :

- Extrait les images conteneur sur Internet
- Crée un déploiement ou un demonset de nœuds, qui fait tourner les pods Astra Trident sur tous les nœuds éligibles du cluster Kubernetes

Installation hors ligne

Le mode d'installation hors ligne peut être requis dans un emplacement rodé ou sécurisé. Dans ce scénario, vous pouvez créer un registre privé en miroir ou deux registres en miroir pour stocker les images Trident et CSI requises.

Quelle que soit la configuration de votre registre, les images CSI doivent résider dans un registre.

Installation à distance

Voici une présentation générale du processus d'installation à distance :

- Déployez la version appropriée de `kubectl` Sur l'ordinateur distant d'où vous souhaitez déployer Astra Trident.
- Copiez les fichiers de configuration depuis le cluster Kubernetes et configurez le `KUBECONFIG` variable d'environnement sur la machine à distance.
- Lancer un `kubectl get nodes` Commande pour vérifier que vous pouvez vous connecter au cluster Kubernetes requis.
- Effectuez le déploiement à partir de la machine distante en suivant les étapes d'installation standard.

Sélectionnez le processus en fonction de votre méthode et de votre mode

Après avoir pris vos décisions, sélectionnez le processus approprié.

Méthode	Mode d'installation
Opérateur Trident (manuellement)	"Installation standard" "Installation hors ligne"
Opérateur Trident (Helm)	"Installation standard" "Installation hors ligne"
`tridentctl`	"Installation standard ou hors ligne"

Passage d'une méthode d'installation à l'autre

Vous pouvez décider de modifier votre méthode d'installation. Avant de procéder, prenez en compte les points suivants :

- Utilisez toujours la même méthode pour installer et désinstaller Astra Trident. Si vous avez déployé avec `tridentctl`, vous devez utiliser la version appropriée de l' `tridentctl` Binaire pour désinstaller Astra Trident. De même, si vous déployez avec l'opérateur, vous devez modifier le `TridentOrchestrator` CR et set `spec.uninstall=true` Pour désinstaller Astra Trident.
- Si vous avez un déploiement basé sur l'opérateur que vous souhaitez supprimer et utiliser à la place `tridentctl` Pour déployer Astra Trident, vous devez d'abord modifier `TridentOrchestrator` et jeu `spec.uninstall=true` Pour désinstaller Astra Trident. Puis supprimer `TridentOrchestrator` et le déploiement de l'opérateur. Vous pouvez ensuite installer à l'aide de `tridentctl`.
- Si vous disposez d'un déploiement manuel basé sur l'opérateur et que vous souhaitez utiliser le déploiement d'opérateurs Trident basé sur Helm, vous devez d'abord désinstaller manuellement l'opérateur, puis effectuer l'installation de Helm. Helm permet à l'opérateur Trident de déployer les étiquettes et les annotations requises. Si vous ne le faites pas, le déploiement d'un opérateur Trident basé sur Helm échoue en raison de l'erreur de validation des étiquettes et de l'erreur de validation des annotations. Si vous avez un `tridentctl`Le déploiement basé sur Helm permet d'utiliser un déploiement basé sur Helm sans s'exécuter dans les problèmes.

Autres options de configuration connues

Lors de l'installation d'Astra Trident sur les produits de la gamme VMware Tanzu :

- Le cluster doit prendre en charge les workloads privilégiés.
- Le `--kubelet-dir` l'indicateur doit être défini sur l'emplacement du répertoire kubelet. Par défaut, il s'agit de `/var/vcap/data/kubelet`.

  Spécifier l'emplacement du kubelet à l'aide de `--kubelet-dir` Est connu pour fonctionner avec l'opérateur Trident, Helm et `tridentctl` de nombreux déploiements.

Installer à l'aide de l'opérateur Trident

Déployer manuellement l'opérateur Trident (mode Standard)

Vous pouvez déployer manuellement l'opérateur Trident pour installer Astra Trident. Ce processus s'applique aux installations dans lesquelles les images de conteneur requises par Astra Trident ne sont pas stockées dans un registre privé. Si vous disposez d'un registre d'images privé, utilisez le "processus de déploiement hors ligne".

Découvrez Astra Trident 23.01, un document essentiel

Vous devez lire les renseignements essentiels suivants sur Astra Trident.

** informations pratiques sur le Tridécouvrez Astra **

- Kubernetes 1.26 est désormais pris en charge par Trident. Mise à niveau de Trident avant la mise à niveau de Kubernetes.
- Astra Trident applique rigoureusement l'utilisation de la configuration de chemins d'accès multiples dans les environnements SAN, avec la valeur recommandée de `find_multipaths: no` dans le fichier multipath.conf.

  Utilisation d'une configuration sans chemins d'accès multiples ou de l'utilisation de `find_multipaths: yes` ou `find_multipaths: smart` la valeur du fichier multipath.conf entraînera des échecs de montage. Trident a recommandé l'utilisation de `find_multipaths: no` depuis la version 21.07.

Déployer manuellement l'opérateur Trident et installer Trident

Révision "présentation de l'installation" pour vous assurer que les conditions préalables à l'installation sont respectées et que vous avez sélectionné l'option d'installation appropriée pour votre environnement.

Avant de commencer

Avant de commencer l'installation, connectez-vous à l'hôte Linux et vérifiez qu'il gère un travail. "Cluster Kubernetes pris en charge" et que vous disposez des privilèges nécessaires.

> Avec OpenShift, utilisez `oc` au lieu de `kubectl` dans tous les exemples qui suivent, et connectez-vous en tant que **system:admin** en premier lieu en cours d'exécution `oc login -u system:admin` ou `oc login -u kube-admin`.

1. Vérifiez votre version Kubernetes :

```
kubectl version
```

2. Vérifiez les privilèges d'administrateur du cluster :

```
kubectl auth can-i '*' '*' --all-namespaces
```

3. Vérifiez que vous pouvez lancer un pod qui utilise une image depuis Docker Hub et atteindre votre système de stockage sur le réseau du pod :

```
kubectl run -i --tty ping --image=busybox --restart=Never --rm -- \
  ping <management IP>
```

Étape 1 : téléchargez le package du programme d'installation de Trident

Le package d'installation d'Astra Trident contient tout ce dont vous avez besoin pour déployer l'opérateur Trident et installer Astra Trident. Téléchargez et extrayez la dernière version du programme d'installation de

Trident "La section *Assets* sur GitHub".

```
wget https://github.com/NetApp/trident/releases/download/v23.01.1/trident-
installer-23.01.1.tar.gz
tar -xf trident-installer-23.01.1.tar.gz
cd trident-installer
```

Étape 2 : créez le `TridentOrchestrator` CRD

Créer le `TridentOrchestrator` Définition de ressource personnalisée (CRD). Vous allez créer un `TridentOrchestrator` Ressources personnalisées plus tard. Utilisez la version CRD YAML appropriée dans `deploy/crds` pour créer le `TridentOrchestrator` CRD.

```
kubectl create -f
deploy/crds/trident.netapp.io_tridentorchestrators_crd_post1.16.yaml
```

Étape 3 : déployer l'opérateur Trident

Le programme d'installation d'Astra Trident fournit un fichier de bundle qui peut être utilisé pour installer l'opérateur et créer des objets associés. Le fichier bundle est un moyen simple de déployer l'opérateur et d'installer Astra Trident avec une configuration par défaut.

- Pour les clusters exécutant Kubernetes 1.24 ou version inférieure, utilisez `bundle_pre_1_25.yaml`.
- Pour les clusters qui exécutent Kubernetes 1.25 ou version supérieure, utilisez `bundle_post_1_25.yaml`.

Le programme d'installation de Trident déploie l'opérateur dans le système `trident` espace de noms. Si le `trident` l'espace de noms n'existe pas, utilisez `kubectl apply -f deploy/namespace.yaml` pour la créer.

Étapes

1. Créer les ressources et déployer l'opérateur :

   ```
   kubectl create -f deploy/<bundle>.yaml
   ```

   > Pour déployer l'opérateur dans un espace de nom autre que le `trident` espace de noms, mettre à jour `serviceaccount.yaml`, `clusterrolebinding.yaml` et `operator.yaml` et générez votre fichier de bundle à l'aide du `kustomization.yaml`:
   >
   > ```
   > kubectl kustomize deploy/ > deploy/<bundle>.yaml
   > ```

2. Vérifier que l'opérateur a été déployé.

```
kubectl get deployment -n <operator-namespace>

NAME                READY   UP-TO-DATE   AVAILABLE   AGE
trident-operator    1/1     1            1           3m
```

Il ne doit y avoir que **une instance** de l'opérateur dans un cluster Kubernetes. Ne créez pas plusieurs déploiements de l'opérateur Trident.

Étape 4 : créez le `TridentOrchestrator` Et installer Trident

Vous pouvez maintenant créer le `TridentOrchestrator` Et installer Astra Trident. Si vous le souhaitez, vous pouvez "Personnalisez votre installation de Trident" utilisation des attributs dans `TridentOrchestrator` spécifications

```
kubectl create -f deploy/crds/tridentorchestrator_cr.yaml
tridentorchestrator.trident.netapp.io/trident created

kubectl describe torc trident

Name:        trident
Namespace:
Labels:      <none>
Annotations: <none>
API Version: trident.netapp.io/v1
Kind:        TridentOrchestrator
...
Spec:
  Debug:     true
  Namespace: trident
Status:
  Current Installation Params:
    IPv6:                      false
    Autosupport Hostname:
    Autosupport Image:         netapp/trident-autosupport:23.01
    Autosupport Proxy:
    Autosupport Serial Number:
    Debug:                     true
    Image Pull Secrets:
    Image Registry:
    k8sTimeout:           30
    Kubelet Dir:          /var/lib/kubelet
    Log Format:           text
    Silence Autosupport:  false
    Trident Image:        netapp/trident:23.01.1
  Message:                  Trident installed  Namespace:
trident
  Status:                   Installed
  Version:                  v23.01.1
Events:
    Type Reason Age From Message ---- ------ ---- ---- -------Normal
    Installing 74s trident-operator.netapp.io Installing Trident Normal
    Installed 67s trident-operator.netapp.io Trident installed
```

Vérifiez l'installation

Il existe plusieurs façons de vérifier votre installation.

À l'aide de `TridentOrchestrator` état

Le statut de `TridentOrchestrator` Indique si l'installation a réussi et affiche la version de Trident installée.

Pendant l'installation, l'état de `TridentOrchestrator` modifications de `Installing` à `Installed`. Si vous observez l' `Failed` l'état et l'opérateur ne parvient pas à récupérer lui-même, "vérifiez les journaux".

État	Description
Installation	L'opérateur installe Astra Trident à l'aide de ce module `TridentOrchestrator` CR.
Installé	Astra Trident a été installé avec succès.
Désinstallation	L'opérateur désinstallant Astra Trident, car `spec.uninstall=true`.
Désinstallé	Astra Trident est désinstallé.
Échec	L'opérateur n'a pas pu installer, corriger, mettre à jour ou désinstaller Astra Trident. L'opérateur essaiera automatiquement de récupérer cet état. Si cet état persiste, vous devrez effectuer un dépannage.
Mise à jour	L'opérateur met à jour une installation existante.
Erreur	Le `TridentOrchestrator` n'est pas utilisé. Un autre existe déjà.

Utilisation du statut de création du pod

Vous pouvez vérifier que l'installation d'Astra Trident est terminée en consultant le statut des pods créés :

```
kubectl get pods -n trident

NAME                                READY   STATUS    RESTARTS
AGE
trident-controller-7d466bf5c7-v4cpw        6/6     Running   0
1m
trident-node-linux-mr6zc                   2/2     Running   0
1m
trident-node-linux-xrp7w                   2/2     Running   0
1m
trident-node-linux-zh2jt                   2/2     Running   0
1m
trident-operator-766f7b8658-ldzsv          1/1     Running   0
3m
```

À l'aide de `tridentctl`

Vous pouvez utiliser `tridentctl` Pour vérifier la version d'Astra Trident installée.

```
./tridentctl -n trident version

+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 23.01.1        | 23.01.1        |
+----------------+----------------+
```

Et la suite

Aujourd'hui c'est possible "création d'une classe de stockage et de back-end, provisionnement d'un volume et montage du volume dans un pod".

Déploiement manuel de l'opérateur Trident (mode hors ligne)

Vous pouvez déployer manuellement l'opérateur Trident pour installer Astra Trident. Ce processus s'applique aux installations dans lesquelles les images de conteneur requises par Astra Trident sont stockées dans un registre privé. Si vous ne disposez pas d'un registre d'images privé, utilisez le "du déploiement standard".

Découvrez Astra Trident 23.01, un document essentiel

Vous devez lire les renseignements essentiels suivants sur Astra Trident.

informations pratiques sur le Tridécouvrez Astra

- Kubernetes 1.26 est désormais pris en charge par Trident. Mise à niveau de Trident avant la mise à niveau de Kubernetes.
- Astra Trident applique rigoureusement l'utilisation de la configuration de chemins d'accès multiples dans les environnements SAN, avec la valeur recommandée de `find_multipaths: no` dans le fichier multipath.conf.

  Utilisation d'une configuration sans chemins d'accès multiples ou de l'utilisation de `find_multipaths: yes` ou `find_multipaths: smart` la valeur du fichier multipath.conf entraînera des échecs de montage. Trident a recommandé l'utilisation de `find_multipaths: no` depuis la version 21.07.

Déployer manuellement l'opérateur Trident et installer Trident

Révision "présentation de l'installation" pour vous assurer que les conditions préalables à l'installation sont respectées et que vous avez sélectionné l'option d'installation appropriée pour votre environnement.

Avant de commencer

Connectez-vous à l'hôte Linux et vérifiez qu'il gère un environnement de travail et "Cluster Kubernetes pris en charge" et que vous disposez des privilèges nécessaires.

Avec OpenShift, utilisez `oc` au lieu de `kubectl` dans tous les exemples qui suivent, et connectez-vous en tant que **system:admin** en premier lieu en cours d'exécution `oc login -u system:admin` ou `oc login -u kube-admin`.

1. Vérifiez votre version Kubernetes :

```
kubectl version
```

2. Vérifiez les privilèges d'administrateur du cluster :

```
kubectl auth can-i '*' '*' --all-namespaces
```

3. Vérifiez que vous pouvez lancer un pod qui utilise une image depuis Docker Hub et atteindre votre système de stockage sur le réseau du pod :

```
kubectl run -i --tty ping --image=busybox --restart=Never --rm -- \
  ping <management IP>
```

Étape 1 : téléchargez le package du programme d'installation de Trident

Le package d'installation d'Astra Trident contient tout ce dont vous avez besoin pour déployer l'opérateur Trident et installer Astra Trident. Téléchargez et extrayez la dernière version du programme d'installation de Trident "La section *Assets* sur GitHub".

```
wget https://github.com/NetApp/trident/releases/download/v23.01.1/trident-
installer-23.01.1.tar.gz
tar -xf trident-installer-23.01.1.tar.gz
cd trident-installer
```

Étape 2 : créez le `TridentOrchestrator` CRD

Créer le `TridentOrchestrator` Définition de ressource personnalisée (CRD). Vous allez créer un `TridentOrchestrator` Ressources personnalisées plus tard. Utilisez la version CRD YAML appropriée dans `deploy/crds` pour créer le `TridentOrchestrator` CRD :

```
kubectl create -f deploy/crds/<VERSION>.yaml
```

Étape 3 : mettez à jour l'emplacement du registre dans l'opérateur

Dans `/deploy/operator.yaml`, mettre à jour `image: docker.io/netapp/trident-operator:23.01.1` pour refléter l'emplacement de votre registre d'images. Votre "Images Trident et CSI"

Peut être situé dans un registre ou dans des registres différents, mais toutes les images CSI doivent se trouver dans le même registre. Par exemple :

- `image: <your-registry>/trident-operator:23.01.1` si vos images sont toutes situées dans un même registre.
- `image: <your-registry>/netapp/trident-operator:23.01.1` Si votre image Trident se trouve dans un registre différent de vos images CSI.

Étape 4 : déploiement de l'opérateur Trident

Le programme d'installation de Trident déploie l'opérateur dans le système `trident` espace de noms. Si le `trident` l'espace de noms n'existe pas, utilisez `kubectl apply -f deploy/namespace.yaml` pour la créer.

Pour déployer l'opérateur dans un espace de nom autre que le `trident` espace de noms, mettre à jour `serviceaccount.yaml`, `clusterrolebinding.yaml` et `operator.yaml` avant de déployer l'opérateur.

1. Créer les ressources et déployer l'opérateur :

```
kubectl kustomize deploy/ > deploy/<BUNDLE>.yaml
```

> Le programme d'installation d'Astra Trident fournit un fichier de bundle qui peut être utilisé pour installer l'opérateur et créer des objets associés. Le fichier bundle est un moyen simple de déployer l'opérateur et d'installer Astra Trident avec une configuration par défaut.
>
> - Pour les clusters exécutant Kubernetes 1.24 ou version inférieure, utilisez `bundle_pre_1_25.yaml`.
> - Pour les clusters qui exécutent Kubernetes 1.25 ou version supérieure, utilisez `bundle_post_1_25.yaml`.

2. Vérifier que l'opérateur a été déployé.

```
kubectl get deployment -n <operator-namespace>

NAME                READY   UP-TO-DATE   AVAILABLE   AGE
trident-operator    1/1     1            1           3m
```

> Il ne doit y avoir que **une instance** de l'opérateur dans un cluster Kubernetes. Ne créez pas plusieurs déploiements de l'opérateur Trident.

Étape 5 : mettez à jour l'emplacement du registre d'images dans le `TridentOrchestrator`

Votre "Images Trident et CSI" Peut être situé dans un registre ou dans des registres différents, mais toutes les images CSI doivent se trouver dans le même registre. Mise à jour `deploy/crds/tridentorchestrator_cr.yaml` pour ajouter les spécifications d'emplacement supplémentaires en fonction de votre configuration de registre.

Images dans un registre

```
imageRegistry: "<your-registry>"
autosupportImage: "<your-registry>/trident-autosupport:23.01"
tridentImage: "<your-registry>/trident:23.01.1"
```

Images dans différents registres

Vous devez ajouter `sig-storage` à la `imageRegistry` pour utiliser différents emplacements de registre.

```
imageRegistry: "<your-registry>/sig-storage"
autosupportImage: "<your-registry>/netapp/trident-autosupport:23.01"
tridentImage: "<your-registry>/netapp/trident:23.01.1"
```

Étape 6 : créez le `TridentOrchestrator` Et installer Trident

Vous pouvez maintenant créer le `TridentOrchestrator` Et installer Astra Trident. Si vous le souhaitez, vous pouvez aussi aller plus loin "Personnalisez votre installation de Trident" utilisation des attributs dans `TridentOrchestrator` spécifications L'exemple suivant montre une installation dans laquelle les images Trident et CSI se trouvent dans différents registres.

```
kubectl create -f deploy/crds/tridentorchestrator_cr.yaml
tridentorchestrator.trident.netapp.io/trident created

kubectl describe torc trident

Name:        trident
Namespace:
Labels:      <none>
Annotations: <none>
API Version: trident.netapp.io/v1
Kind:        TridentOrchestrator
...
Spec:
  Autosupport Image:  <your-registry>/netapp/trident-autosupport:23.01
  Debug:              true
  Image Registry:     <your-registry>/sig-storage
  Namespace:          trident
  Trident Image:      <your-registry>/netapp/trident:23.01.1
Status:
  Current Installation Params:
    IPv6:                       false
    Autosupport Hostname:
    Autosupport Image:          <your-registry>/netapp/trident-
autosupport:23.01
    Autosupport Proxy:
    Autosupport Serial Number:
    Debug:                      true
    Http Request Timeout:       90s
    Image Pull Secrets:
    Image Registry:       <your-registry>/sig-storage
    k8sTimeout:           30
    Kubelet Dir:          /var/lib/kubelet
    Log Format:           text
    Probe Port:           17546
    Silence Autosupport:  false
    Trident Image:        <your-registry>/netapp/trident:23.01.1
  Message:                Trident installed
  Namespace:              trident
  Status:                 Installed
  Version:                v23.01.1
Events:
    Type Reason Age From Message ---- ------ ---- ---- -------Normal
    Installing 74s trident-operator.netapp.io Installing Trident Normal
    Installed 67s trident-operator.netapp.io Trident installed
```

Vérifiez l'installation

Il existe plusieurs façons de vérifier votre installation.

À l'aide de `TridentOrchestrator` état

Le statut de `TridentOrchestrator` Indique si l'installation a réussi et affiche la version de Trident installée. Pendant l'installation, l'état de `TridentOrchestrator` modifications de `Installing` à `Installed`. Si vous observez l' `Failed` l'état et l'opérateur ne parvient pas à récupérer lui-même, "vérifiez les journaux".

État	Description
Installation	L'opérateur installe Astra Trident à l'aide de ce module `TridentOrchestrator` CR.
Installé	Astra Trident a été installé avec succès.
Désinstallation	L'opérateur désinstallant Astra Trident, car `spec.uninstall=true`.
Désinstallé	Astra Trident est désinstallé.
Échec	L'opérateur n'a pas pu installer, corriger, mettre à jour ou désinstaller Astra Trident. L'opérateur essaiera automatiquement de récupérer cet état. Si cet état persiste, vous devrez effectuer un dépannage.
Mise à jour	L'opérateur met à jour une installation existante.
Erreur	Le `TridentOrchestrator` n'est pas utilisé. Un autre existe déjà.

Utilisation du statut de création du pod

Vous pouvez vérifier que l'installation d'Astra Trident est terminée en consultant le statut des pods créés :

```
kubectl get pods -n trident

NAME                                READY   STATUS    RESTARTS
AGE
trident-controller-7d466bf5c7-v4cpw        6/6     Running   0
1m
trident-node-linux-mr6zc                   2/2     Running   0
1m
trident-node-linux-xrp7w                   2/2     Running   0
1m
trident-node-linux-zh2jt                   2/2     Running   0
1m
trident-operator-766f7b8658-ldzsv          1/1     Running   0
3m
```

À l'aide de `tridentctl`

Vous pouvez utiliser `tridentctl` Pour vérifier la version d'Astra Trident installée.

```
./tridentctl -n trident version

+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 23.01.1        | 23.01.1        |
+----------------+----------------+
```

Et la suite

Aujourd'hui c'est possible "création d'une classe de stockage et de back-end, provisionnement d'un volume et montage du volume dans un pod".

Déploiement de l'opérateur Trident à l'aide de Helm (mode standard)

Vous pouvez déployer l'opérateur Trident et installer Astra Trident à l'aide de Helm. Ce processus s'applique aux installations dans lesquelles les images de conteneur requises par Astra Trident ne sont pas stockées dans un registre privé. Si vous disposez d'un registre d'images privé, utilisez le "processus de déploiement hors ligne".

Découvrez Astra Trident 23.01, un document essentiel

Vous devez lire les renseignements essentiels suivants sur Astra Trident.

informations pratiques sur le Tridécouvrez Astra

- Kubernetes 1.26 est désormais pris en charge par Trident. Mise à niveau de Trident avant la mise à niveau de Kubernetes.
- Astra Trident applique rigoureusement l'utilisation de la configuration de chemins d'accès multiples dans les environnements SAN, avec la valeur recommandée de `find_multipaths: no` dans le fichier multipath.conf.

  Utilisation d'une configuration sans chemins d'accès multiples ou de l'utilisation de `find_multipaths: yes` ou `find_multipaths: smart` la valeur du fichier multipath.conf entraînera des échecs de montage. Trident a recommandé l'utilisation de `find_multipaths: no` depuis la version 21.07.

Déployer l'opérateur Trident et installer Astra Trident à l'aide de Helm

Avec Trident "Graphique Helm" Vous pouvez déployer l'opérateur Trident et installer Trident en une étape.

Révision "présentation de l'installation" pour vous assurer que les conditions préalables à l'installation sont respectées et que vous avez sélectionné l'option d'installation appropriée pour votre environnement.

Avant de commencer

En plus du "conditions préalables au déploiement" dont vous avez besoin "Version 3 de Helm".

Étapes

1. Ajout du référentiel Astra Trident Helm :

```
helm repo add netapp-trident https://netapp.github.io/trident-helm-chart
```

2. Utiliser `helm install` et spécifiez un nom pour votre déploiement comme dans l'exemple suivant où `23.01.1` Est la version d'Astra Trident que vous installez.

```
helm install <name> netapp-trident/trident-operator --version 23.01.1
--create-namespace --namespace <trident-namespace>
```

> Si vous avez déjà créé un namespace pour Trident, le `--create-namespace` le paramètre ne crée pas d'espace de noms supplémentaire.

Vous pouvez utiliser `helm list` pour vérifier les détails de l'installation tels que le nom, l'espace de noms, le graphique, l'état, la version de l'application, et numéro de révision.

Transmettre les données de configuration pendant l'installation

Il existe deux façons de passer les données de configuration au cours de l'installation :

Option	Description
`--values` (ou `-f`)	Spécifiez un fichier YAML avec les remplacements. Ceci peut être spécifié plusieurs fois et le fichier le plus à droite sera prioritaire.
`--set`	Spécifiez les remplacements sur la ligne de commande.

Par exemple, pour modifier la valeur par défaut de `debug`, exécutez ce qui suit `--set` commande où `23.01.1` Est la version d'Astra Trident que vous installez :

```
helm install <name> netapp-trident/trident-operator --version 23.01.1
--create-namespace --namespace --set tridentDebug=true
```

Options de configuration

Ce tableau et le `values.yaml` Le fichier, qui fait partie du graphique Helm, fournit la liste des clés et leurs valeurs par défaut.

Option	Description	Valeur par défaut
`nodeSelector`	Libellés des nœuds pour l'affectation des pods	
`podAnnotations`	Annotations de pod	
`deploymentAnnotations`	Annotations de déploiement	
`tolerations`	Tolérances pour l'affectation de pod	
`affinity`	Affinité pour l'affectation de pod	
`tridentControllerPluginNodeSelector`	Sélecteurs de nœuds supplémentaires pour les pods. Reportez-vous à la section Présentation des pods de contrôleur et des nœuds pour plus d'informations.	
`tridentControllerPluginTolerations`	Remplace les tolérances Kubernetes pour les pods. Reportez-vous à la section Présentation des pods de contrôleur et des nœuds pour plus d'informations.	
`tridentNodePluginNodeSelector`	Sélecteurs de nœuds supplémentaires pour les pods. Reportez-vous à la section Présentation des pods de contrôleur et des nœuds pour plus d'informations.	
`tridentNodePluginTolerations`	Remplace les tolérances Kubernetes pour les pods. Reportez-vous à la section Présentation des pods de contrôleur et des nœuds pour plus d'informations.	
`imageRegistry`	Identifie le registre du `trident-operator`, `trident`, et autres images. Laissez vide pour accepter la valeur par défaut.	« »
`imagePullPolicy`	Définit la stratégie d'extraction d'image pour le `trident-operator`.	`IfNotPresent`
`imagePullSecrets`	Définit les secrets d'extraction d'image pour le `trident-operator`, `trident`, et autres images.	
`kubeletDir`	Permet de remplacer l'emplacement hôte de l'état interne du kubelet.	"'/var/lib/kubelet"'

Option	Description	Valeur par défaut
`operatorLogLevel`	Permet de définir le niveau du journal de l'opérateur Trident sur : `trace`, `debug`, `info`, `warn`, `error`, ou `fatal`.	`"info"`
`operatorDebug`	Permet de définir le niveau du journal de l'opérateur Trident sur DEBUG.	`true`
`operatorImage`	Permet la neutralisation complète de l'image pour `trident-operator`.	« »
`operatorImageTag`	Permet de remplacer la balise du `trident-operator` image.	« »
`tridentIPv6`	Permet à Astra Trident de fonctionner dans des clusters IPv6.	`false`
`tridentK8sTimeout`	Remplace le délai d'expiration par défaut de 30 secondes pour la plupart des opérations de l'API Kubernetes (s'il n'est pas égal à zéro, en secondes).	`0`
`tridentHttpRequestTimeout`	Remplace le délai par défaut de 90 secondes pour les requêtes HTTP, par `0s` étant une durée infinie pour le délai d'expiration. Les valeurs négatives ne sont pas autorisées.	`"90s"`
`tridentSilenceAutosupport`	Permet de désactiver le reporting AutoSupport périodique d'Astra Trident.	`false`
`tridentAutosupportImageTag`	Permet de remplacer la balise de l'image pour le conteneur AutoSupport Astra Trident.	`<version>`
`tridentAutosupportProxy`	Permet au conteneur AutoSupport Astra Trident de Phone Home via un proxy HTTP.	« »
`tridentLogFormat`	Définit le format de journalisation d'Astra Trident (`text` ou `json`).	`"text"`
`tridentDisableAuditLog`	Désactive l'enregistreur d'audit Astra Trident.	`true`
`tridentLogLevel`	Permet de définir le niveau de journal d'Astra Trident sur : `trace`, `debug`, `info`, `warn`, `error`, ou `fatal`.	`"info"`
`tridentDebug`	Permet de définir le niveau du journal d'Astra Trident sur `debug`.	`false`

Option	Description	Valeur par défaut
`tridentLogWorkflows`	Permet d'activer des workflows Astra Trident spécifiques pour la journalisation des traces ou la suppression de journaux.	« »
`tridentLogLayers`	Permet d'activer des couches Astra Trident spécifiques pour la journalisation des traces ou la suppression des journaux.	« »
`tridentImage`	Permet la neutralisation complète de l'image pour Astra Trident.	« »
`tridentImageTag`	Permet de remplacer la balise de l'image pour Astra Trident.	« »
`tridentProbePort`	Permet de remplacer le port par défaut utilisé pour les sondes de disponibilité/préparation Kubernetes.	« »
`windows`	Permet d'installer Astra Trident sur le nœud de travail Windows.	`false`
`enableForceDetach`	Permet d'activer la fonction forcer le détachement.	`false`
`excludePodSecurityPolicy`	Exclut la stratégie de sécurité du module opérateur de la création.	`false`

Présentation des pods de contrôleur et des nœuds

ASTRA Trident s'exécute comme un seul pod de contrôleur, plus un pod de nœud sur chaque nœud worker dans le cluster. Le pod de nœuds doit être exécuté sur n'importe quel hôte sur lequel vous souhaitez potentiellement monter un volume Astra Trident.

Kubernetes "sélecteurs de nœuds" et "tolérances et rejets" sont utilisés pour contraindre un pod à s'exécuter sur un nœud spécifique ou préféré. En utilisant le « ControllerPlugin » et `NodePlugin`, vous pouvez spécifier des contraintes et des remplacements.

- Le plug-in du contrôleur gère le provisionnement et la gestion des volumes, tels que les snapshots et le redimensionnement.
- Le plug-in du nœud permet d'attacher le stockage au nœud.

Et la suite

Aujourd'hui c'est possible "création d'une classe de stockage et de back-end, provisionnement d'un volume et montage du volume dans un pod".

Déploiement de l'opérateur Trident à l'aide de Helm (mode hors ligne)

Vous pouvez déployer l'opérateur Trident et installer Astra Trident à l'aide de Helm. Ce processus s'applique aux installations dans lesquelles les images de conteneur requises par Astra Trident sont stockées dans un registre privé. Si vous ne disposez pas d'un

registre d'images privé, utilisez le "du déploiement standard".

Découvrez Astra Trident 23.01, un document essentiel

Vous devez lire les renseignements essentiels suivants sur Astra Trident.

** informations pratiques sur le Tridécouvrez Astra **

- Kubernetes 1.26 est désormais pris en charge par Trident. Mise à niveau de Trident avant la mise à niveau de Kubernetes.
- Astra Trident applique rigoureusement l'utilisation de la configuration de chemins d'accès multiples dans les environnements SAN, avec la valeur recommandée de `find_multipaths: no` dans le fichier multipath.conf.

  Utilisation d'une configuration sans chemins d'accès multiples ou de l'utilisation de `find_multipaths: yes` ou `find_multipaths: smart` la valeur du fichier multipath.conf entraînera des échecs de montage. Trident a recommandé l'utilisation de `find_multipaths: no` depuis la version 21.07.

Déployer l'opérateur Trident et installer Astra Trident à l'aide de Helm

Avec Trident "Graphique Helm" Vous pouvez déployer l'opérateur Trident et installer Trident en une étape.

Révision "présentation de l'installation" pour vous assurer que les conditions préalables à l'installation sont respectées et que vous avez sélectionné l'option d'installation appropriée pour votre environnement.

Avant de commencer

En plus du "conditions préalables au déploiement" dont vous avez besoin "Version 3 de Helm".

Étapes

1. Ajout du référentiel Astra Trident Helm :

   ```
   helm repo add netapp-trident https://netapp.github.io/trident-helm-chart
   ```

2. Utiliser `helm install` et spécifiez un nom pour votre déploiement et l'emplacement du registre d'images. Votre "Images Trident et CSI" Peut être situé dans un registre ou dans des registres différents, mais toutes les images CSI doivent se trouver dans le même registre. Dans les exemples, `23.01.1` Est la version d'Astra Trident que vous installez.

Images dans un registre

```
helm install <name> netapp-trident/trident-operator --version
23.01.1 --set imageRegistry=<your-registry> --create-namespace
--namespace <trident-namespace>
```

Images dans différents registres

Vous devez ajouter `sig-storage` à la `imageRegistry` pour utiliser différents emplacements de registre.

```
helm install <name> netapp-trident/trident-operator --version
23.01.1 --set imageRegistry=<your-registry>/sig-storage --set
operatorImage=<your-registry>/netapp/trident-operator:23.01.1 --set
tridentAutosupportImage=<your-registry>/netapp/trident-
autosupport:23.01 --set tridentImage=<your-
registry>/netapp/trident:23.01.1 --create-namespace --namespace
<trident-namespace>
```

Si vous avez déjà créé un namespace pour Trident, le `--create-namespace` le paramètre ne crée pas d'espace de noms supplémentaire.

Vous pouvez utiliser `helm list` pour vérifier les détails de l'installation tels que le nom, l'espace de noms, le graphique, l'état, la version de l'application, et numéro de révision.

Transmettre les données de configuration pendant l'installation

Il existe deux façons de passer les données de configuration au cours de l'installation :

Option	Description
`--values` (ou `-f`)	Spécifiez un fichier YAML avec les remplacements. Ceci peut être spécifié plusieurs fois et le fichier le plus à droite sera prioritaire.
`--set`	Spécifiez les remplacements sur la ligne de commande.

Par exemple, pour modifier la valeur par défaut de `debug`, exécutez ce qui suit `--set` commande où `23.01.1` Est la version d'Astra Trident que vous installez :

```
helm install <name> netapp-trident/trident-operator --version 23.01.1
--create-namespace --namespace --set tridentDebug=true
```

Options de configuration

Ce tableau et le `values.yaml` Le fichier, qui fait partie du graphique Helm, fournit la liste des clés et leurs valeurs par défaut.

Option	Description	Valeur par défaut
`nodeSelector`	Libellés des nœuds pour l'affectation des pods	
`podAnnotations`	Annotations de pod	
`deploymentAnnotations`	Annotations de déploiement	
`tolerations`	Tolérances pour l'affectation de pod	
`affinity`	Affinité pour l'affectation de pod	
`tridentControllerPluginNodeSelector`	Sélecteurs de nœuds supplémentaires pour les pods. Reportez-vous à la section Présentation des pods de contrôleur et des nœuds pour plus d'informations.	
`tridentControllerPluginTolerations`	Remplace les tolérances Kubernetes pour les pods. Reportez-vous à la section Présentation des pods de contrôleur et des nœuds pour plus d'informations.	
`tridentNodePluginNodeSelector`	Sélecteurs de nœuds supplémentaires pour les pods. Reportez-vous à la section Présentation des pods de contrôleur et des nœuds pour plus d'informations.	
`tridentNodePluginTolerations`	Remplace les tolérances Kubernetes pour les pods. Reportez-vous à la section Présentation des pods de contrôleur et des nœuds pour plus d'informations.	
`imageRegistry`	Identifie le registre du `trident-operator`, `trident`, et autres images. Laissez vide pour accepter la valeur par défaut.	« »
`imagePullPolicy`	Définit la stratégie d'extraction d'image pour le `trident-operator`.	`IfNotPresent`
`imagePullSecrets`	Définit les secrets d'extraction d'image pour le `trident-operator`, `trident`, et autres images.	

Option	Description	Valeur par défaut
`kubeletDir`	Permet de remplacer l'emplacement hôte de l'état interne du kubelet.	`"/var/lib/kubelet"`
`operatorLogLevel`	Permet de définir le niveau du journal de l'opérateur Trident sur : `trace`, `debug`, `info`, `warn`, `error`, ou `fatal`.	`"info"`
`operatorDebug`	Permet de définir le niveau du journal de l'opérateur Trident sur DEBUG.	`true`
`operatorImage`	Permet la neutralisation complète de l'image pour `trident-operator`.	« »
`operatorImageTag`	Permet de remplacer la balise du `trident-operator` image.	« »
`tridentIPv6`	Permet à Astra Trident de fonctionner dans des clusters IPv6.	`false`
`tridentK8sTimeout`	Remplace le délai d'expiration par défaut de 30 secondes pour la plupart des opérations de l'API Kubernetes (s'il n'est pas égal à zéro, en secondes).	`0`
`tridentHttpRequestTimeout`	Remplace le délai par défaut de 90 secondes pour les requêtes HTTP, par `0s` étant une durée infinie pour le délai d'expiration. Les valeurs négatives ne sont pas autorisées.	`"90s"`
`tridentSilenceAutosupport`	Permet de désactiver le reporting AutoSupport périodique d'Astra Trident.	`false`
`tridentAutosupportImageTag`	Permet de remplacer la balise de l'image pour le conteneur AutoSupport Astra Trident.	`<version>`
`tridentAutosupportProxy`	Permet au conteneur AutoSupport Astra Trident de Phone Home via un proxy HTTP.	« »
`tridentLogFormat`	Définit le format de journalisation d'Astra Trident (`text` ou `json`).	`"text"`
`tridentDisableAuditLog`	Désactive l'enregistreur d'audit Astra Trident.	`true`
`tridentLogLevel`	Permet de définir le niveau de journal d'Astra Trident sur : `trace`, `debug`, `info`, `warn`, `error`, ou `fatal`.	`"info"`

Option	Description	Valeur par défaut
`tridentDebug`	Permet de définir le niveau du journal d'Astra Trident sur `debug`.	`false`
`tridentLogWorkflows`	Permet d'activer des workflows Astra Trident spécifiques pour la journalisation des traces ou la suppression de journaux.	« »
`tridentLogLayers`	Permet d'activer des couches Astra Trident spécifiques pour la journalisation des traces ou la suppression des journaux.	« »
`tridentImage`	Permet la neutralisation complète de l'image pour Astra Trident.	« »
`tridentImageTag`	Permet de remplacer la balise de l'image pour Astra Trident.	« »
`tridentProbePort`	Permet de remplacer le port par défaut utilisé pour les sondes de disponibilité/préparation Kubernetes.	« »
`windows`	Permet d'installer Astra Trident sur le nœud de travail Windows.	`false`
`enableForceDetach`	Permet d'activer la fonction forcer le détachement.	`false`
`excludePodSecurityPolicy`	Exclut la stratégie de sécurité du module opérateur de la création.	`false`

Présentation des pods de contrôleur et des nœuds

ASTRA Trident s'exécute comme un seul pod de contrôleur, plus un pod de nœud sur chaque nœud worker dans le cluster. Le pod de nœuds doit être exécuté sur n'importe quel hôte sur lequel vous souhaitez potentiellement monter un volume Astra Trident.

Kubernetes "sélecteurs de nœuds" et "tolérances et rejets" sont utilisés pour contraindre un pod à s'exécuter sur un nœud spécifique ou préféré. En utilisant le « ControllerPlugin » et `NodePlugin`, vous pouvez spécifier des contraintes et des remplacements.

- Le plug-in du contrôleur gère le provisionnement et la gestion des volumes, tels que les snapshots et le redimensionnement.
- Le plug-in du nœud permet d'attacher le stockage au nœud.

Et la suite

Aujourd'hui c'est possible "création d'une classe de stockage et de back-end, provisionnement d'un volume et montage du volume dans un pod".

Personnalisez l'installation de l'opérateur Trident

L'opérateur Trident vous permet de personnaliser l'installation d'Astra Trident à l'aide des

attributs du `TridentOrchestrator` spécifications Si vous voulez personnaliser l'installation au-delà de ce qui est `TridentOrchestrator` les arguments permettent, envisagez d'utiliser `tridentctl` Pour générer des manifestes YAML personnalisés à modifier selon les besoins.

Présentation des pods de contrôleur et des nœuds

ASTRA Trident s'exécute comme un seul pod de contrôleur, plus un pod de nœud sur chaque nœud worker dans le cluster. Le pod de nœuds doit être exécuté sur n'importe quel hôte sur lequel vous souhaitez potentiellement monter un volume Astra Trident.

Kubernetes "sélecteurs de nœuds" et "tolérances et rejets" sont utilisés pour contraindre un pod à s'exécuter sur un nœud spécifique ou préféré. En utilisant le « ControllerPlugin » et `NodePlugin`, vous pouvez spécifier des contraintes et des remplacements.

- Le plug-in du contrôleur gère le provisionnement et la gestion des volumes, tels que les snapshots et le redimensionnement.
- Le plug-in du nœud permet d'attacher le stockage au nœud.

Options de configuration

`spec.namespace` est spécifié dans `TridentOrchestrator` Pour indiquer l'espace de noms dans lequel Astra Trident est installé. Ce paramètre **ne peut pas être mis à jour après l'installation d'Astra Trident**. Pour tenter de le faire, le `TridentOrchestrator` statut pour passer à `Failed`. Astra Trident n'est pas conçu pour être migré entre les espaces de noms.

Ce tableau est plus détaillé `TridentOrchestrator` attributs.

Paramètre	Description	Valeur par défaut
`namespace`	Espace de noms pour installer Astra Trident dans	« par défaut »
`debug`	Activez le débogage pour Astra Trident	faux
`windows`	Réglage sur `true` Active l'installation sur les nœuds de travail Windows.	faux
`IPv6`	Installez Astra Trident sur IPv6	faux
`k8sTimeout`	Délai d'expiration pour les opérations Kubernetes	30 secondes
`silenceAutosupport`	N'envoyez pas automatiquement des packs AutoSupport à NetApp	faux
`enableNodePrep`	Gérer automatiquement les dépendances des nœuds de travail (**BÊTA**)	faux
`autosupportImage`	Image conteneur pour la télémétrie AutoSupport	netapp/trident-autosupport:23.01

Paramètre	Description	Valeur par défaut
`autosupportProxy`	Adresse/port d'un proxy pour l'envoi de télémétrie AutoSupport	"http://proxy.example.com:8888"
`uninstall`	Indicateur utilisé pour désinstaller Astra Trident	faux
`logFormat`	Format de connexion Astra Trident à utiliser [text,json]	« texte »
`tridentImage`	Image Astra Trident à installer	netapp/trident:21.04
`imageRegistry`	Chemin d'accès au registre interne, du format `<registry FQDN>[:port][/subpath]`	"k8s.gcr.io/sig-storage (k8s 1.19+) ou quay.io/k8scsi"
`kubeletDir`	Chemin d'accès au répertoire kubelet de l'hôte	"/var/lib/kubelet"
`wipeout`	Liste des ressources à supprimer pour effectuer la suppression complète d'Astra Trident	
`imagePullSecrets`	Secrets pour extraire des images d'un registre interne	
`imagePullPolicy`	Définit la stratégie de collecte d'image pour l'opérateur Trident. Les valeurs valides sont : `Always` pour toujours tirer l'image. `IfNotPresent` pour extraire l'image uniquement s'il n'existe pas déjà sur le nœud. `Never` pour ne jamais tirer l'image.	`IfNotPresent`
`controllerPluginNodeSelector`	Sélecteurs de nœuds supplémentaires pour les pods. Suit le même format que pod.spec.nodeSelector.	Pas de valeur par défaut ; facultatif
`controllerPluginTolerations`	Remplace les tolérances Kubernetes pour les pods. Suit le même format que pod.spec.tolérances.	Pas de valeur par défaut ; facultatif
`nodePluginNodeSelector`	Sélecteurs de nœuds supplémentaires pour les pods. Suit le même format que pod.spec.nodeSelector.	Pas de valeur par défaut ; facultatif
`nodePluginTolerations`	Remplace les tolérances Kubernetes pour les pods. Suit le même format que pod.spec.tolérances.	Pas de valeur par défaut ; facultatif

Pour plus d'informations sur le formatage des paramètres du pod, reportez-vous à la section "Attribution de pods aux nœuds".

Exemples de configurations

Vous pouvez utiliser les attributs mentionnés ci-dessus lors de la définition `TridentOrchestrator` pour personnaliser votre installation.

Exemple 1 : configuration personnalisée de base

Voici un exemple de configuration personnalisée de base.

```
cat deploy/crds/tridentorchestrator_cr_imagepullsecrets.yaml
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  imagePullSecrets:
  - thisisasecret
```

Exemple 2 : déploiement avec des sélecteurs de nœuds

Cet exemple illustre le déploiement de Trident avec des sélecteurs de nœud :

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  controllerPluginNodeSelector:
    nodetype: master
  nodePluginNodeSelector:
    storage: netapp
```

Exemple 3 : déploiement sur des nœuds de travail Windows

Cet exemple illustre le déploiement sur un nœud de travail Windows.

```
cat deploy/crds/tridentorchestrator_cr.yaml
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  windows: true
```

Installation à l'aide de tridentctl

Installation à l'aide de tridentctl

Vous pouvez installer Astra Trident à l'aide de `tridentctl`. Ce processus s'applique aux installations où les images de conteneur requises par Astra Trident sont stockées dans un registre privé ou non. Pour personnaliser votre `tridentctl` déploiement, voir "Personnalisez le déploiement tridentctl".

Découvrez Astra Trident 23.01, un document essentiel

Vous devez lire les renseignements essentiels suivants sur Astra Trident.

informations pratiques sur le Tridécouvrez Astra

- Kubernetes 1.26 est désormais pris en charge par Trident. Mise à niveau de Trident avant la mise à niveau de Kubernetes.
- Astra Trident applique rigoureusement l'utilisation de la configuration de chemins d'accès multiples dans les environnements SAN, avec la valeur recommandée de `find_multipaths: no` dans le fichier multipath.conf.

  Utilisation d'une configuration sans chemins d'accès multiples ou de l'utilisation de `find_multipaths: yes` ou `find_multipaths: smart` la valeur du fichier multipath.conf entraînera des échecs de montage. Trident a recommandé l'utilisation de `find_multipaths: no` depuis la version 21.07.

Installation d'Astra Trident à l'aide de `tridentctl`

Révision "présentation de l'installation" pour vous assurer que les conditions préalables à l'installation sont respectées et que vous avez sélectionné l'option d'installation appropriée pour votre environnement.

Avant de commencer

Avant de commencer l'installation, connectez-vous à l'hôte Linux et vérifiez qu'il gère un travail. "Cluster Kubernetes pris en charge" et que vous disposez des privilèges nécessaires.

> Avec OpenShift, utilisez `oc` au lieu de `kubectl` dans tous les exemples qui suivent, et connectez-vous en tant que **system:admin** en premier lieu en cours d'exécution `oc login -u system:admin` ou `oc login -u kube-admin`.

1. Vérifiez votre version Kubernetes :

```
kubectl version
```

2. Vérifiez les privilèges d'administrateur du cluster :

```
kubectl auth can-i '*' '*' --all-namespaces
```

3. Vérifiez que vous pouvez lancer un pod qui utilise une image depuis Docker Hub et atteindre votre système de stockage sur le réseau du pod :

```
kubectl run -i --tty ping --image=busybox --restart=Never --rm -- \
  ping <management IP>
```

Étape 1 : téléchargez le package du programme d'installation de Trident

Le package du programme d'installation Astra Trident crée un pod Trident, configure les objets CRD utilisés pour maintenir son état et initialise les sidecars CSI pour effectuer des actions telles que le provisionnement et la connexion de volumes aux hôtes du cluster. Téléchargez et extrayez la dernière version du programme d'installation de Trident "La section *Assets* sur GitHub". Mettez à jour *<trident-installer-XX.XX.X.tar.gz>* dans l'exemple avec la version Trident Astra que vous avez sélectionnée.

```
wget https://github.com/NetApp/trident/releases/download/v23.01.1/trident-
installer-23.01.1.tar.gz
tar -xf trident-installer-23.01.1.tar.gz
cd trident-installer
```

Étape 2 : installer Astra Trident

Installez Astra Trident dans l'espace de noms souhaité en exécutant le `tridentctl install` commande. Vous pouvez ajouter des arguments supplémentaires pour spécifier l'emplacement du registre d'images.

> Pour permettre à Astra Trident de s'exécuter sur des nœuds Windows, ajoutez le `--windows` indicateur de la commande install : `$ ./tridentctl install --windows -n trident`.

Mode standard

```
./tridentctl install -n trident
```

Images dans un registre

```
./tridentctl install -n trident --image-registry <your-registry>
--autosupport-image <your-registry>/trident-autosupport:23.01 --trident
-image <your-registry>/trident:23.01.1
```

Images dans différents registres

Vous devez ajouter `sig-storage` à la `imageRegistry` pour utiliser différents emplacements de registre.

```
./tridentctl install -n trident --image-registry <your-registry>/sig-
storage --autosupport-image <your-registry>/netapp/trident-
autosupport:23.01 --trident-image <your-
registry>/netapp/trident:23.01.1
```

L'état de votre installation devrait ressembler à ceci.

```
....
INFO Starting Trident installation.                namespace=trident
INFO Created service account.
INFO Created cluster role.
INFO Created cluster role binding.
INFO Added finalizers to custom resource definitions.
INFO Created Trident service.
INFO Created Trident secret.
INFO Created Trident deployment.
INFO Created Trident daemonset.
INFO Waiting for Trident pod to start.
INFO Trident pod started.                          namespace=trident
pod=trident-controller-679648bd45-cv2mx
INFO Waiting for Trident REST interface.
INFO Trident REST interface is up.                 version=23.01.1
INFO Trident installation succeeded.
....
```

Vérifiez l'installation

Vous pouvez vérifier votre installation à l'aide de l'état de création du pod ou `tridentctl`.

Utilisation du statut de création du pod

Vous pouvez vérifier que l'installation d'Astra Trident est terminée en consultant le statut des pods créés :

```
kubectl get pods -n trident

NAME                                READY   STATUS    RESTARTS   AGE
trident-controller-679648bd45-cv2mx   6/6     Running   0          5m29s
trident-node-linux-vgc8n              2/2     Running   0          5m29s
```

Si le programme d'installation ne s'est pas terminé correctement ou `trident-controller-<generated id>` (`trident-csi-<generated id>` Dans les versions antérieures à 23.01) n'ont pas d'état **en cours d'exécution**, la plate-forme n'était pas installée. Utiliser `-d` à "activer le mode débogage" et de résoudre le problème.

À l'aide de `tridentctl`

Vous pouvez utiliser `tridentctl` Pour vérifier la version d'Astra Trident installée.

```
./tridentctl -n trident version

+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 23.01.1        | 23.01.1        |
+----------------+----------------+
```

Et la suite

Aujourd'hui c'est possible "création d'une classe de stockage et de back-end, provisionnement d'un volume et montage du volume dans un pod".

Personnalisez l'installation tridentctl

Vous pouvez utiliser le programme d'installation d'Astra Trident pour personnaliser l'installation.

En savoir plus sur le programme d'installation

Le programme d'installation d'Astra Trident vous permet de personnaliser les attributs. Par exemple, si vous avez copié l'image Trident dans un référentiel privé, vous pouvez spécifier le nom de l'image à l'aide de `--trident-image`. Si vous avez copié l'image Trident ainsi que les images sidecar CSI nécessaires dans un référentiel privé, il est peut-être préférable de spécifier l'emplacement de ce référentiel à l'aide du `--image-registry` commutateur, qui prend la forme `<registry FQDN>[:port]`.

Si vous utilisez une distribution de Kubernetes, où `kubelet` conserve ses données sur un chemin différent de la normale `/var/lib/kubelet`, vous pouvez spécifier la trajectoire alternative en utilisant `--kubelet-dir`.

Si vous devez personnaliser l'installation au-delà de ce que les arguments du programme d'installation autorisent, vous pouvez également personnaliser les fichiers de déploiement. À l'aide du `--generate-custom-yaml` Le paramètre crée les fichiers YAML suivants dans le programme d'installation `setup` répertoire :

- `trident-clusterrolebinding.yaml`
- `trident-deployment.yaml`
- `trident-crds.yaml`
- `trident-clusterrole.yaml`
- `trident-daemonset.yaml`
- `trident-service.yaml`
- `trident-namespace.yaml`
- `trident-serviceaccount.yaml`
- `trident-resourcequota.yaml`

Après avoir généré ces fichiers, vous pouvez les modifier en fonction de vos besoins, puis les utiliser `--use-custom-yaml` pour installer votre déploiement personnalisé.

```
./tridentctl install -n trident --use-custom-yaml
```

Et la suite ?

Une fois Astra Trident installé, vous pouvez créer un système back-end, créer une classe de stockage, provisionner un volume et monter le volume dans un pod.

Étape 1 : créer un back-end

Vous pouvez à présent créer un système back-end qui sera utilisé par Astra Trident pour provisionner des volumes. Pour ce faire, créez un `backend.json` fichier contenant les paramètres nécessaires. Des exemples de fichiers de configuration pour différents types backend sont disponibles dans le `sample-input` répertoire.

Voir "ici" pour plus de détails sur la configuration du fichier pour votre type backend.

```
cp sample-input/<backend template>.json backend.json
vi backend.json
```

```
./tridentctl -n trident create backend -f backend.json
+-------------+---------------+--------------------------------------
+--------+---------+
|    NAME     | STORAGE DRIVER |                 UUID                 |
STATE   | VOLUMES |
+-------------+---------------+--------------------------------------
+--------+---------+
| nas-backend | ontap-nas      | 98e19b74-aec7-4a3d-8dcf-128e5033b214 |
online |       0 |
+-------------+---------------+--------------------------------------
+--------+---------+
```

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

```
./tridentctl -n trident logs
```

Une fois que vous avez résolu le problème, revenez tout simplement au début de cette étape et réessayez. Pour plus de conseils de dépannage, reportez-vous à la section "le dépannage" section.

Étape 2 : créer une classe de stockage

Les utilisateurs Kubernetes provisionnent des volumes à l'aide de demandes de volume persistant qui spécifient un volume "classe de stockage" par nom. Les détails sont masqués des utilisateurs, mais une classe de stockage identifie le mécanisme de provisionnement utilisé pour cette classe (dans ce cas, Trident), et ce que cette classe signifie pour le mécanisme de provisionnement.

Créez une classe de stockage que les utilisateurs Kubernetes spécifient quand ils veulent un volume. La configuration de la classe doit modéliser le back-end que vous avez créé à l'étape précédente, de sorte qu'Astra Trident l'utilise pour provisionner de nouveaux volumes.

Pour commencer, la classe de stockage la plus simple est basée sur la `sample-input/storage-class-csi.yaml.templ` fichier fourni avec le programme d'installation, remplacement *`BACKEND_TYPE`* avec le nom du pilote de stockage.

```
./tridentctl -n trident get backend
+-------------+---------------+--------------------------------------
+--------+---------+
|    NAME     | STORAGE DRIVER |                 UUID                 |
STATE  | VOLUMES |
+-------------+---------------+--------------------------------------
+--------+---------+
| nas-backend | ontap-nas      | 98e19b74-aec7-4a3d-8dcf-128e5033b214 |
online |       0 |
+-------------+---------------+--------------------------------------
+--------+---------+

cp sample-input/storage-class-csi.yaml.templ sample-input/storage-class-
basic-csi.yaml

# Modify __BACKEND_TYPE__ with the storage driver field above (e.g.,
ontap-nas)
vi sample-input/storage-class-basic-csi.yaml
```

Il s'agit d'un objet Kubernetes, que vous utilisez `kubectl` Pour la créer dans Kubernetes.

```
kubectl create -f sample-input/storage-class-basic-csi.yaml
```

Vous devriez désormais voir une classe de stockage **Basic-csi** dans Kubernetes et Astra Trident. Astra Trident devrait avoir découvert les pools sur le système back-end.

```
kubectl get sc basic-csi
NAME         PROVISIONER             AGE
basic-csi    csi.trident.netapp.io   15h

./tridentctl -n trident get storageclass basic-csi -o json
{
  "items": [
    {
      "Config": {
        "version": "1",
        "name": "basic-csi",
        "attributes": {
          "backendType": "ontap-nas"
        },
        "storagePools": null,
        "additionalStoragePools": null
      },
      "storage": {
        "ontapnas_10.0.0.1": [
          "aggr1",
          "aggr2",
          "aggr3",
          "aggr4"
        ]
      }
    }
  ]
}
```

Étape 3 : provisionner le premier volume

Vous êtes désormais prêt à provisionner votre premier volume de façon dynamique. Pour ce faire, vous créez un environnement Kubernetes "demande de volume persistant" (PVC) objet.

Créez un volume persistant pour un volume qui utilise la classe de stockage que vous venez de créer.

Voir `sample-input/pvc-basic-csi.yaml` par exemple. Assurez-vous que le nom de la classe de stockage correspond à celui que vous avez créé.

```
kubectl create -f sample-input/pvc-basic-csi.yaml

kubectl get pvc --watch
NAME      STATUS    VOLUME                                     CAPACITY
ACCESS MODES  STORAGECLASS   AGE
basic     Pending
basic          1s
basic     Pending   pvc-3acb0d1c-b1ae-11e9-8d9f-5254004dfdb7   0
basic          5s
basic     Bound     pvc-3acb0d1c-b1ae-11e9-8d9f-5254004dfdb7   1Gi
RWO           basic          7s
```

Étape 4 : montez les volumes dans un pod

Examinons maintenant le volume. Nous allons lancer un module nginx qui monte le PV sous `/usr/share/nginx/html`.

```
cat << EOF > task-pv-pod.yaml
kind: Pod
apiVersion: v1
metadata:
  name: task-pv-pod
spec:
  volumes:
    - name: task-pv-storage
      persistentVolumeClaim:
       claimName: basic
  containers:
    - name: task-pv-container
      image: nginx
      ports:
        - containerPort: 80
          name: "http-server"
      volumeMounts:
        - mountPath: "/usr/share/nginx/html"
          name: task-pv-storage
EOF
kubectl create -f task-pv-pod.yaml
```

```
# Wait for the pod to start
kubectl get pod --watch

# Verify that the volume is mounted on /usr/share/nginx/html
kubectl exec -it task-pv-pod -- df -h /usr/share/nginx/html

# Delete the pod
kubectl delete pod task-pv-pod
```

À ce stade, le pod (application) n'existe plus, mais le volume est toujours là. Vous pouvez l'utiliser à partir d'un autre pod si vous le souhaitez.

Pour supprimer le volume, supprimez la réclamation :

```
kubectl delete pvc basic
```

Vous pouvez désormais effectuer d'autres tâches, telles que :

- "Configuration des systèmes back-end supplémentaires"
- "Créer des classes de stockage supplémentaires."

Gérez Astra Trident

Mettez à niveau Astra Trident

Mettez à niveau Astra Trident

Astra Trident suit le rythme de sa mise à jour trimestrielle, fournissant quatre versions majeures chaque année. Chaque nouvelle version s'appuie sur les versions précédentes, offrant de nouvelles fonctionnalités et améliorations des performances ainsi que des correctifs et des améliorations. Nous vous encourageons à effectuer une mise à niveau au moins une fois par an pour profiter des nouvelles fonctionnalités d'Astra Trident.

Sélectionnez une version

Les versions d'Astra Trident suivent une date `YY.MM` convention de dénomination, où "YY" est les deux derniers chiffres de l'année et "MM" est le mois. Les versions point suivent un `YY.MM.X` convention, où « X » est le niveau de patch. Vous allez sélectionner la version à mettre à niveau en fonction de la version à partir de laquelle vous effectuez la mise à niveau.

- Vous pouvez effectuer une mise à niveau directe vers n'importe quelle version cible située dans une fenêtre à quatre versions de la version installée. Par exemple, vous pouvez effectuer la mise à niveau vers 23.01 à partir de 22.01 (y compris les versions point, telles que 22.01.1) directement.
- Si vous disposez d'une version antérieure, vous devez effectuer une mise à niveau en plusieurs étapes à l'aide de la documentation de la version concernée pour obtenir des instructions spécifiques. Pour ce faire, vous devez d'abord effectuer une mise à niveau vers la version la plus récente qui correspond à votre fenêtre des quatre versions. Par exemple, si vous exécutez 18.07 et que vous souhaitez effectuer une mise à niveau vers la version 20.07, suivez la procédure de mise à niveau en plusieurs étapes comme suit :
  a. Première mise à niveau de 18.07 à 19.07.
  b. Puis mettre à niveau de 19.07 à 20.07.

> - Toutes les mises à niveau vers les versions 19.04 et précédentes exigent la migration des métadonnées Astra Trident `etcd` Aux objets CRD. Assurez-vous de consulter la documentation de la version pour comprendre le fonctionnement de la mise à niveau.
> - Lors de la mise à niveau, il est important de fournir `parameter.fsType` dans `StorageClasses` Utilisé par Astra Trident. Vous pouvez supprimer et recréer `StorageClasses` sans interrompre les volumes existants. Il s'agit d'une **exigence** pour appliquer security contextes pour les volumes SAN. Le répertoire sample input contient des exemples, tels que `storage-class-basic.yaml.templ` et `storage-class-bronze-default.yaml`. Pour plus d'informations, voir "Problèmes connus".

Sélectionnez une option de mise à niveau

Il existe deux options de mise à niveau d'Astra Trident. En général, vous utiliserez la même option que celle que vous avez utilisée pour l'installation initiale, mais vous le pouvez "passer d'une méthode d'installation à l'autre".

- "Mise à niveau à l'aide de l'opérateur Trident"
  *

Les snapshots de volumes CSI sont désormais une fonctionnalité GA, qui commence par Kubernetes 1.20. Lors de la mise à niveau d'Astra Trident, tous les précédents clichés alpha CRS et CRD (classes Snapshot de volume, instantanés de volume et contenu Snapshot de volume) doivent être supprimés avant la mise à niveau. Reportez-vous à la section "de ce blog" Comprendre les étapes de migration des instantanés alpha vers les spécifications bêta/GA.

Modifications apportées à l'opérateur

La version 21.01 d'Astra Trident présente quelques changements architecturaux clés pour l'opérateur, à savoir :

- L'opérateur est maintenant **cluster-scoped**. Les instances précédentes de l'opérateur Trident (versions 20.04 à 20.10) étaient **namespace-scoped**. Un opérateur à périmètre de bloc d'instruments est avantageux pour les raisons suivantes :
  - Responsabilité des ressources : l'opérateur gère désormais les ressources associées à une installation d'Astra Trident au niveau du cluster. Dans le cadre de l'installation d'Astra Trident, l'opérateur crée et gère plusieurs ressources à l'aide de `ownerReferences`. Maintenance `ownerReferences` Sur les ressources cluster-scoped peut générer des erreurs sur certains distributeurs Kubernetes tels qu'OpenShift. Ceci est réduit avec un opérateur à périmètre sur le cluster. Pour l'auto-rétablissement et l'application de correctifs des ressources Trident, c'est une condition essentielle.
  - Nettoyage pendant la désinstallation : une suppression complète d'Astra Trident nécessite la suppression de toutes les ressources associées. Un opérateur de type espace de noms peut rencontrer des problèmes lors de la suppression des ressources du cluster (telles que clusterRole, ClusterRoleBinding et PodSecurityPolicy) et entraîner un nettoyage incomplet. Un opérateur à périmètre de cluster élimine ce problème. Les utilisateurs peuvent désinstaller complètement Astra Trident et procéder à un nouveau battage si nécessaire.
- `TridentProvisioner` est maintenant remplacé par `TridentOrchestrator` Ressource personnalisée utilisée pour installer et gérer Astra Trident. En outre, un nouveau champ est introduit dans le `TridentOrchestrator` spécifications Les utilisateurs peuvent spécifier que le namespace Trident doit être installé/mis à niveau à partir du à l'aide du `spec.namespace` légale. Vous pouvez voir un exemple "ici".

Mise à niveau avec l'opérateur

Vous pouvez facilement mettre à niveau une installation Astra Trident existante à l'aide de l'opérateur.

Avant de commencer

Pour effectuer la mise à niveau à l'aide de l'opérateur, les conditions suivantes doivent être remplies :

- Vous devez disposer d'une installation Astra Trident basée sur CSI. Toutes les versions de 19.07 sont basées sur CSI. Vous pouvez examiner les pods dans votre espace de noms Trident pour en vérifier l'état.
  - Dans les versions antérieures à 23.01, le nom des pods suit un `trident-csi-*` convention.
  - Le nom de pod dans la version 23.01 et ultérieures utilise : `trident-controller-<generated id>` pour le pod de contrôleurs ; `trident-node-<operating system>-<generated id>` pour les pods des nœuds ; `trident-operator-<generated id>` pour le boîtier de commande.
- Si vous avez désinstallé CSI Trident et que les métadonnées de l'installation persistent, vous pouvez effectuer une mise à niveau à l'aide de l'opérateur.
- Une seule installation Astra Trident doit exister sur l'ensemble des espaces de noms d'un cluster

Kubernetes donné.

- Vous devez utiliser un cluster Kubernetes en cours d'exécution "Version Kubernetes prise en charge".
- Si des CRD alpha snapshot sont présents, vous devez les supprimer avec `tridentctl obliviate alpha-snapshot-crd`. Ceci supprime les CRD pour la spécification de snapshot alpha. Pour les snapshots existants qui doivent être supprimés/migrés, reportez-vous à la section "de ce blog".

> - Lors de la mise à niveau de Trident à l'aide de l'opérateur sur OpenShift Container Platform, nous vous recommandons d'effectuer une mise à niveau vers Trident 21.01.1 ou version ultérieure. L'opérateur Trident sorti avec 21.01.0 contient un problème connu qui a été résolu en 21.01.1. Pour plus de détails, consultez le "Consultez le document GitHub pour plus d'informations".
> - N'utilisez pas l'opérateur pour mettre à niveau Trident si vous utilisez un `etcd`Version Trident basée (19.04 ou version antérieure).

Mettre à niveau l'installation d'un opérateur Trident du type cluster

Procédez comme suit pour mettre à niveau l'installation d'un opérateur Trident du système d'extension du cluster. Toutes les versions 21.01 et supérieures d'Astra Trident utilisent un opérateur à périmètre de cluster.

Étapes

1. Vérifiez votre version d'Astra Trident :

```
./tridentctl -n trident version
```

2. Supprimez l'opérateur Trident qui a été utilisé pour installer l'instance Astra Trident actuelle. Par exemple, si vous effectuez une mise à niveau depuis 22.01, exécutez la commande suivante :

```
kubectl delete -f 22.01/trident-installer/deploy/bundle.yaml -n trident
```

3. Si vous avez personnalisé votre installation initiale à l'aide de `TridentOrchestrator` attributs, vous pouvez modifier le `TridentOrchestrator` objet pour modifier les paramètres d'installation. Cela peut inclure des modifications visant à spécifier les registres d'images en miroir Trident et CSI pour le mode hors ligne, à activer les journaux de débogage ou à spécifier les secrets d'extraction d'images.
4. Installez Astra Trident à l'aide du fichier YAML correspondant à votre environnement et à la version Trident d'Astra. Par exemple, si vous installez Astra Trident 23.01 pour Kubernetes 1.26, exécutez la commande suivante :

```
kubectl create -f 23.01.1/trident-installer/deploy/bundle_post_1_25.yaml
-n trident
```

Trident fournit un fichier bundle qui peut être utilisé pour installer l'opérateur et créer les objets associés pour votre version Kubernetes.

- Pour les clusters exécutant Kubernetes 1.24 ou version inférieure, utilisez "bundle_pre_1_25.yaml".
- Pour les clusters qui exécutent Kubernetes 1.25 ou version supérieure, utilisez "bundle_post_1_25.yaml".

Résultats

L'opérateur de Trident identifiera une installation Astra Trident existante et la mettra à niveau vers la même version que l'opérateur.

Mettre à niveau l'installation d'un opérateur à l'aide d'un espace de noms

Suivez ces étapes pour effectuer une mise à niveau à partir d'une instance d'Astra Trident installée à l'aide de l'opérateur doté de l'espace de noms (versions 20.07 à 20.10).

Étapes

1. Vérifiez l'état de l'installation Trident existante. Pour ce faire, vérifiez le **Statut** de `TridentProvisioner`. Le statut doit être de `Installed`.

```
kubectl describe tprov trident -n trident | grep Message: -A 3
Message:  Trident installed
Status:   Installed
Version:  v20.10.1
```

Si l'état s'affiche `Updating`, assurez-vous de le résoudre avant de continuer. Pour obtenir la liste des valeurs d'état possibles, reportez-vous à la section "ici".

2. Créer le `TridentOrchestrator` CRD en utilisant le manifeste fourni avec le programme d'installation Trident.

```
# Download the release required [23.01.1]
mkdir 23.01.1
cd 23.01.1
wget
https://github.com/NetApp/trident/releases/download/v23.01.1/trident-
installer-23.01.1.tar.gz
tar -xf trident-installer-23.01.1.tar.gz
cd trident-installer
kubectl create -f
deploy/crds/trident.netapp.io_tridentorchestrators_crd_post1.16.yaml
```

3. Supprimez l'opérateur délimité par l'espace de noms à l'aide de son manifeste. Pour effectuer cette étape, vous devez utiliser le fichier YAML de bundle pour déployer l'opérateur à étendue de l'espace de noms à partir de `https://github.com/NetApp/trident/tree/stable/vXX.XX/deploy/BUNDLE.YAML`

où *vXX.XX* est le numéro de version et *BUNDLE.YAML* Est le nom du fichier YAML du bundle.

> Vous devez apporter les modifications nécessaires aux paramètres d'installation de Trident (par exemple, en modifiant les valeurs de `tridentImage`, `autosupportImage`, un référentiel d'images privé, et la fourniture `imagePullSecrets`) après avoir supprimé l'opérateur du périmètre de l'espace de noms et avant d'installer l'opérateur du périmètre de cluster. Pour obtenir une liste complète des paramètres pouvant être mis à jour, reportez-vous au "options de configuration".

```
#Ensure you are in the right directory
pwd
/root/20.10.1/trident-installer

#Delete the namespace-scoped operator
kubectl delete -f deploy/<BUNDLE.YAML> -n trident
serviceaccount "trident-operator" deleted
clusterrole.rbac.authorization.k8s.io "trident-operator" deleted
clusterrolebinding.rbac.authorization.k8s.io "trident-operator" deleted
deployment.apps "trident-operator" deleted
podsecuritypolicy.policy "tridentoperatorpods" deleted

#Confirm the Trident operator was removed
kubectl get all -n trident
NAME                                  READY   STATUS    RESTARTS   AGE
pod/trident-csi-68d979fb85-dsrmn   6/6     Running   12         99d
pod/trident-csi-8jfhf              2/2     Running   6          105d
pod/trident-csi-jtnjz              2/2     Running   6          105d
pod/trident-csi-lcxvh              2/2     Running   8          105d

NAME                 TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)
AGE
service/trident-csi   ClusterIP   10.108.174.125   <none>
34571/TCP,9220/TCP   105d

NAME                           DESIRED   CURRENT   READY   UP-TO-DATE
AVAILABLE   NODE SELECTOR                                        AGE
daemonset.apps/trident-csi   3         3         3       3            3
kubernetes.io/arch=amd64,kubernetes.io/os=linux   105d

NAME                           READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/trident-csi   1/1     1            1           105d

NAME                                     DESIRED   CURRENT   READY   AGE
replicaset.apps/trident-csi-68d979fb85   1         1         1
105d
```

À ce stade, le `trident-operator-xxxxxxxxxx-xxxxx` le pod a été supprimé.

4. (Facultatif) si les paramètres d'installation doivent être modifiés, mettez à jour le `TridentProvisioner` spécifications Il peut s'agir de modifications telles que la modification du registre d'images privées pour extraire des images de conteneur, l'activation des journaux de débogage ou la spécification de secrets de collecte d'images.

```
kubectl patch tprov <trident-provisioner-name> -n <trident-namespace>
--type=merge -p '{"spec":{"debug":true}}'
```

5. Installez l'opérateur Trident.

> L'installation de l'opérateur à périmètre de cluster initie la migration de `TridentProvisioner` objets à `TridentOrchestrator` objets, supprime `TridentProvisioner` objets et le `tridentprovisioner` CRD, et met à niveau Astra Trident vers la version de l'opérateur délimité par le cluster. Dans l'exemple qui suit, Trident est mis à niveau vers la version 23.01.1.

> La mise à niveau d'Astra Trident avec l'opérateur Trident entraîne la migration de `tridentProvisioner` à un `tridentOrchestrator` objet portant le même nom. Cette opération est gérée automatiquement par l'opérateur. La mise à niveau entraînera également l'installation d'Astra Trident dans le même espace de noms qu'auparavant.

```
#Ensure you are in the correct directory
pwd
/root/23.01.1/trident-installer

#Install the cluster-scoped operator in the **same namespace**
kubectl create -f deploy/<BUNDLE.YAML>
serviceaccount/trident-operator created
clusterrole.rbac.authorization.k8s.io/trident-operator created
clusterrolebinding.rbac.authorization.k8s.io/trident-operator created
deployment.apps/trident-operator created
podsecuritypolicy.policy/tridentoperatorpods created

#All tridentProvisioners will be removed, including the CRD itself
kubectl get tprov -n trident
Error from server (NotFound): Unable to list "trident.netapp.io/v1,
Resource=tridentprovisioners": the server could not find the requested
resource (get tridentprovisioners.trident.netapp.io)

#tridentProvisioners are replaced by tridentOrchestrator
kubectl get torc
NAME      AGE
trident   13s

#Examine Trident pods in the namespace
kubectl get pods -n trident
NAME                                READY   STATUS    RESTARTS
AGE
trident-controller-79df798bdc-m79dc      6/6     Running   0
1m41s
trident-node-linux-xrst8                 2/2     Running   0
1m41s
trident-operator-5574dbbc68-nthjv        1/1     Running   0
1m52s

#Confirm Trident has been updated to the desired version
kubectl describe torc trident | grep Message -A 3
Message:                Trident installed
Namespace:              trident
Status:                 Installed
Version:                v23.01.1
```

Le `trident-controller` les noms de pods reflètent la convention de nommage introduite en 23.01.

Mettre à niveau l'installation d'un opérateur basé sur Helm

Effectuer les étapes suivantes pour mettre à niveau l'installation d'un opérateur reposant sur Helm.

> Lorsque vous mettez à niveau un cluster Kubernetes de 1.24 vers 1.25 ou version ultérieure sur lequel Astra Trident est installé, vous devez mettre à jour les valeurs.yaml pour les définir `excludePodSecurityPolicy` à `true` ou ajouter `--set excludePodSecurityPolicy=true` à la `helm upgrade` commande avant de pouvoir mettre à niveau le cluster.

Étapes

1. Téléchargez la dernière version d'Astra Trident.
2. Utilisez le `helm upgrade` commande où `trident-operator-23.01.1.tgz` reflète la version vers laquelle vous souhaitez effectuer la mise à niveau.

```
helm upgrade <name> trident-operator-23.01.1.tgz
```

> Si vous définissez des options autres que celles par défaut lors de l'installation initiale (par exemple, spécifier des registres privés en miroir pour les images Trident et CSI), utilisez `--set` pour vous assurer que ces options sont incluses dans la commande de mise à niveau, sinon les valeurs sont réinitialisées sur les valeurs par défaut.
>
> Par exemple, pour modifier la valeur par défaut de `tridentDebug`, exécutez la commande suivante :
>
> ```
> helm upgrade <name> trident-operator-23.01.1-custom.tgz --set
> tridentDebug=true
> ```

3. Courez `helm list` pour vérifier que le graphique et la version de l'application ont tous deux été mis à niveau. Courez `tridentctl logs` pour consulter les messages de débogage.

Résultats

L'opérateur de Trident identifiera une installation Astra Trident existante et la mettra à niveau vers la même version que l'opérateur.

Mise à niveau à partir d'une installation autre que celle d'un opérateur

Vous pouvez effectuer la mise à niveau vers la dernière version de l'opérateur Trident à partir d'un `tridentctl` installation.

Étapes

1. Téléchargez la dernière version d'Astra Trident.

```
# Download the release required [23.01.1]
mkdir 23.01.1
cd 23.01.1
wget
https://github.com/NetApp/trident/releases/download/v22.01.1/trident-
installer-23.01.1.tar.gz
tar -xf trident-installer-23.01.1.tar.gz
cd trident-installer
```

2. Créer le `tridentorchestrator` CRD du manifeste.

```
kubectl create -f
deploy/crds/trident.netapp.io_tridentorchestrators_crd_post1.16.yaml
```

3. Déployer l'opérateur.

```
#Install the cluster-scoped operator in the **same namespace**
kubectl create -f deploy/<BUNDLE.YAML>
serviceaccount/trident-operator created
clusterrole.rbac.authorization.k8s.io/trident-operator created
clusterrolebinding.rbac.authorization.k8s.io/trident-operator created
deployment.apps/trident-operator created
podsecuritypolicy.policy/tridentoperatorpods created

#Examine the pods in the Trident namespace
NAME                                READY   STATUS    RESTARTS   AGE
trident-controller-79df798bdc-m79dc   6/6     Running   0          150d
trident-node-linux-xrst8              2/2     Running   0          150d
trident-operator-5574dbbc68-nthjv     1/1     Running   0          1m30s
```

4. Créer un `TridentOrchestrator` CR pour l'installation d'Astra Trident.

```
#Create a tridentOrchestrator to initiate a Trident install
cat deploy/crds/tridentorchestrator_cr.yaml
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident

kubectl create -f deploy/crds/tridentorchestrator_cr.yaml

#Examine the pods in the Trident namespace
NAME                                READY   STATUS    RESTARTS   AGE
trident-csi-79df798bdc-m79dc        6/6     Running   0          1m
trident-csi-xrst8                   2/2     Running   0          1m
trident-operator-5574dbbc68-nthjv   1/1     Running   0          5m41s

#Confirm Trident was upgraded to the desired version
kubectl describe torc trident | grep Message -A 3
Message:                Trident installed
Namespace:              trident
Status:                 Installed
Version:                v23.01.1
```

Résultats

Les systèmes back-end et demandes de volume persistant sont automatiquement disponibles.

Mise à niveau avec tridentctl

Vous pouvez facilement mettre à niveau une installation Astra Trident existante à l'aide de `tridentctl`.

Considérations avant la mise à niveau

Lorsque vous mettez à niveau vers la dernière version d'Astra Trident, prenez en compte les points suivants :

- Depuis Trident 20.01, seule la version bêta de "snapshots de volume" est pris en charge. Les administrateurs Kubernetes doivent veiller à sauvegarder ou convertir en version bêta les objets de snapshot alpha en toute sécurité, afin de conserver les snapshots alpha hérités.
- La version bêta de snapshots de volume introduit un ensemble modifié de CRD et un contrôleur de snapshot, qui doivent tous deux être configurés avant d'installer Astra Trident. "De ce blog" décrit les étapes de migration des instantanés de volume alpha vers le format bêta.
- La désinstallation et la réinstallation d'Astra Trident fait office de mise à niveau. Lorsque vous désinstallez Trident, la demande de volume persistant et le volume persistant utilisés par l'Astra Trident. Les volumes persistants ayant déjà été provisionnés restent disponibles pendant la mise hors ligne d'Astra Trident, et

Astra Trident provisionne les volumes pour les demandes de volume persistant créées dans l'intervalle une fois de nouveau en ligne.

> Pour la mise à niveau d'Astra Trident, n'interrompez pas le processus. Assurez-vous que le programme d'installation s'exécute jusqu'à la fin.

Étapes suivantes après la mise à niveau

Pour utiliser le riche ensemble de fonctionnalités disponibles dans les dernières versions de Trident (par exemple, les copies Snapshot de volume à la demande), vous pouvez mettre à niveau les volumes à l'aide du `tridentctl upgrade` commande.

S'il existe des volumes hérités, il est conseillé de les mettre à niveau d'un type NFS/iSCSI vers un type CSI pour pouvoir utiliser l'ensemble des nouvelles fonctionnalités d'Astra Trident. Un volume persistant existant provisionné par Trident prend en charge l'ensemble classique de fonctionnalités.

Tenez compte des éléments suivants lorsque vous décidez de mettre à niveau des volumes vers le type CSI :

- Vous n'avez peut-être pas besoin de mettre à niveau tous les volumes. Les volumes déjà créés continuent à être accessibles et fonctionneront normalement.
- Un PV peut être monté dans le cadre d'un déploiement/StatefulSet lors de la mise à niveau. Il n'est pas nécessaire de faire descendre le déploiement/StatefulSet.
- Vous **ne pouvez pas** connecter un PV à un pod autonome lors de la mise à niveau. Vous devez arrêter le pod avant de mettre à niveau le volume.
- Vous pouvez mettre à niveau uniquement un volume lié à un volume persistant. Les volumes qui ne sont pas liés à des demandes de volume persistant doivent être supprimés et importés avant la mise à niveau.

Exemple de mise à niveau de volume

Voici un exemple illustrant le mode d'exécution d'une mise à niveau de volume.

1. Courez `kubectl get pv` Pour répertorier les volumes persistants.

```
kubectl get pv
NAME                        CAPACITY     ACCESS MODES   RECLAIM POLICY
STATUS   CLAIM                   STORAGECLASS    REASON   AGE
default-pvc-1-a8475         1073741824   RWO            Delete
Bound    default/pvc-1          standard                19h
default-pvc-2-a8486         1073741824   RWO            Delete
Bound    default/pvc-2          standard                19h
default-pvc-3-a849e         1073741824   RWO            Delete
Bound    default/pvc-3          standard                19h
default-pvc-4-a84de         1073741824   RWO            Delete
Bound    default/pvc-4          standard                19h
trident                     2Gi          RWO            Retain
Bound    trident/trident                                19h
```

Actuellement, quatre volumes persistants ont été créés par Trident 20.07, à l'aide de `netapp.io/trident` provisionnement.

2. Courez `kubectl describe pv` Pour en savoir plus sur le volume persistant.

```
kubectl describe pv default-pvc-2-a8486

Name:            default-pvc-2-a8486
Labels:          <none>
Annotations:     pv.kubernetes.io/provisioned-by: netapp.io/trident
                 volume.beta.kubernetes.io/storage-class: standard
Finalizers:      [kubernetes.io/pv-protection]
StorageClass:    standard
Status:          Bound
Claim:           default/pvc-2
Reclaim Policy:  Delete
Access Modes:    RWO
VolumeMode:      Filesystem
Capacity:        1073741824
Node Affinity:   <none>
Message:
Source:
    Type:      NFS (an NFS mount that lasts the lifetime of a pod)
    Server:    10.xx.xx.xx
    Path:      /trid_1907_alpha_default_pvc_2_a8486
    ReadOnly:  false
```

Le volume persistant a été créé à l'aide du `netapp.io/trident` de type provisionnement et en mode NFS. Pour prendre en charge toutes les nouvelles fonctionnalités proposées par Astra Trident, ce PV doit être mis à niveau vers le type CSI.

3. Exécutez le `tridentctl upgrade volume <name-of-trident-volume>` Commande de mise à niveau d'un volume Astra Trident hérité vers les spécifications CSI.

```
./tridentctl get volumes -n trident
+---------------------+---------+---------------+----------
+--------------------------------------+--------+---------+
|            NAME     |  SIZE   | STORAGE CLASS | PROTOCOL |
BACKEND UUID             | STATE  | MANAGED |
+---------------------+---------+---------------+----------
+--------------------------------------+--------+---------+
| default-pvc-2-a8486 | 1.0 GiB | standard      | file     | c5a6f6a4-
b052-423b-80d4-8fb491a14a22 | online | true    |
| default-pvc-3-a849e | 1.0 GiB | standard      | file     | c5a6f6a4-
b052-423b-80d4-8fb491a14a22 | online | true    |
| default-pvc-1-a8475 | 1.0 GiB | standard      | file     | c5a6f6a4-
b052-423b-80d4-8fb491a14a22 | online | true    |
| default-pvc-4-a84de | 1.0 GiB | standard      | file     | c5a6f6a4-
b052-423b-80d4-8fb491a14a22 | online | true    |
+---------------------+---------+---------------+----------
+--------------------------------------+--------+---------+

./tridentctl upgrade volume default-pvc-2-a8486 -n trident
+---------------------+---------+---------------+----------
+--------------------------------------+--------+---------+
|            NAME     |  SIZE   | STORAGE CLASS | PROTOCOL |
BACKEND UUID             | STATE  | MANAGED |
+---------------------+---------+---------------+----------
+--------------------------------------+--------+---------+
| default-pvc-2-a8486 | 1.0 GiB | standard      | file     | c5a6f6a4-
b052-423b-80d4-8fb491a14a22 | online | true    |
+---------------------+---------+---------------+----------
+--------------------------------------+--------+---------+
```

4. Exécutez un `kubectl describe pv` Pour vérifier que le volume est un volume CSI.

```
kubectl describe pv default-pvc-2-a8486
Name:            default-pvc-2-a8486
Labels:          <none>
Annotations:     pv.kubernetes.io/provisioned-by: csi.trident.netapp.io
                 volume.beta.kubernetes.io/storage-class: standard
Finalizers:      [kubernetes.io/pv-protection]
StorageClass:    standard
Status:          Bound
Claim:           default/pvc-2
Reclaim Policy:  Delete
Access Modes:    RWO
VolumeMode:      Filesystem
Capacity:        1073741824
Node Affinity:   <none>
Message:
Source:
    Type:              CSI (a Container Storage Interface (CSI) volume
source)
    Driver:            csi.trident.netapp.io
    VolumeHandle:      default-pvc-2-a8486
    ReadOnly:          false
    VolumeAttributes:      backendUUID=c5a6f6a4-b052-423b-80d4-
8fb491a14a22

internalName=trid_1907_alpha_default_pvc_2_a8486
                           name=default-pvc-2-a8486
                           protocol=file
Events:                <none>
```

Ainsi, vous pouvez mettre à niveau des volumes de type NFS/iSCSI créés par Astra Trident vers un type CSI, sur la base du volume.

Désinstaller Astra Trident

Selon l'installation d'Astra Trident, il existe plusieurs options pour le désinstaller.

Désinstaller en utilisant Helm

Si vous avez installé Astra Trident à l'aide de Helm, vous pouvez le désinstaller à l'aide de `helm uninstall`.

```
#List the Helm release corresponding to the Astra Trident install.
helm ls -n trident
NAME          NAMESPACE       REVISION        UPDATED
STATUS          CHART                           APP VERSION
trident       trident         1               2021-04-20
00:26:42.417764794 +0000 UTC deployed        trident-operator-21.07.1
21.07.1

#Uninstall Helm release to remove Trident
helm uninstall trident -n trident
release "trident" uninstalled
```

Désinstaller en utilisant l'opérateur Trident

Si vous avez installé Astra Trident à l'aide de l'opérateur, vous pouvez le désinstaller en procédant de l'une des manières suivantes :

- **Modifier `TridentOrchestrator` Pour définir l'indicateur de désinstallation :**, vous pouvez modifier `TridentOrchestrator` et jeu `spec.uninstall=true`. Modifiez le `TridentOrchestrator` CR et régler le `uninstall` marquer comme indiqué ci-dessous :

  ```
  kubectl patch torc <trident-orchestrator-name> --type=merge -p
  '{"spec":{"uninstall":true}}'
  ```

  Lorsque le `uninstall` l'indicateur est défini sur `true`, L'opérateur Trident désinstalle Trident, mais ne supprime pas TridentOrchestrator lui-même. Vous devez nettoyer TridentOrchestrator et en créer un nouveau si vous souhaitez réinstaller Trident.

- **Supprimer `TridentOrchestrator`:** en retirant le `TridentOrchestrator` CR utilisé pour déployer Astra Trident, vous demandez à l'opérateur de désinstaller Trident. L'opérateur traite la dépose de `TridentOrchestrator` Il procède également au retrait du déploiement et de la demonset Astra Trident, en supprimant les pods Trident qu'il avait créés dans le cadre de l'installation. Pour supprimer entièrement Astra Trident (y compris les CRD qu'il crée) et nettoyer efficacement la ardoise, vous pouvez la modifier `TridentOrchestrator` pour passer `wipeout` option. Voir l'exemple suivant :

```
kubectl patch torc <trident-orchestrator-name> --type=merge -p
'{"spec":{"wipeout":["crds"],"uninstall":true}}'
```

Cela désinstalle complètement Astra Trident et efface toutes les métadonnées relatives aux systèmes back-end et aux volumes gérés. Les installations ultérieures sont traitées comme des installations neuves.

> Vous ne devez envisager de supprimer les CRD que lorsque vous effectuez une désinstallation complète. Cette opération ne peut pas être annulée. **Ne nettoyez pas les CRD à moins que vous ne cherchiez à recommencer et à créer une nouvelle installation Astra Trident**.

Désinstaller à l'aide de `tridentctl`

Exécutez le `uninstall` commande dans `tridentctl` Comme suit, supprime toutes les ressources associées à Astra Trident, à l'exception des CRD et des objets associés, ce qui facilite l'exécution du programme d'installation pour la mise à jour vers une version plus récente.

```
./tridentctl uninstall -n <namespace>
```

Pour supprimer totalement Astra Trident, il est conseillé de supprimer les finaliseurs des CRD créés par Astra Trident et de supprimer les CRD.

Revenir à la version antérieure d'Astra Trident

Découvrez les étapes de la restauration d'une version antérieure d'Astra Trident.

Quand revenir à une version antérieure

Vous pouvez envisager de rétrograder pour diverses raisons, comme les suivantes :

- Planification des mesures d'urgence
- Résolution immédiate des bugs observés après une mise à niveau
- Problèmes de dépendance, mises à niveau infructueuses et incomplètes

Vous devez envisager une version antérieure si vous passez à une version d'Astra Trident qui utilise des CRD. Comme Astra Trident utilise des CRD pour la maintenance de l'état, toutes les entités de stockage créées (systèmes back-end, classes de stockage, volumes persistants et copies de volume) ont associé des objets CRD au lieu de données écrites dans le `trident` PV (utilisé par la version installée précédente d'Astra Trident). Les nouveaux volumes persistants, systèmes back-end et classes de stockage sont tous gérés en tant qu'objets CRD.

Ne tentez de revenir à une version antérieure d'Astra Trident qui s'exécute avec des CRD (19.07 et versions ultérieures). Cela permet de s'assurer que les opérations effectuées sur la version actuelle d'Astra Trident sont visibles après la mise à niveau vers le bas.

Si vous ne devez pas revenir à une version antérieure

Vous ne devez pas revenir à une version antérieure de Trident utilisée `etcd` pour conserver l'état (19.04 et versions antérieures). Toutes les opérations effectuées avec la version actuelle d'Astra Trident ne sont pas prises en compte après la mise à niveau vers le bas. Les nouveaux volumes persistants ne sont pas utilisables lors du déplacement vers une version antérieure. Les modifications apportées aux objets tels que les systèmes back-end, les volumes persistants, les classes de stockage et les copies Snapshot de volume (créées/mises à jour/supprimées) ne sont pas visibles pour Astra Trident lors du retour dans une version antérieure. Revenir à une version antérieure ne perturbe pas l'accès aux volumes persistants créés à l'aide de cette version antérieure, sauf si ces derniers ont été mis à niveau.

Procédure de rétrogradation lorsque Astra Trident est installé à l'aide de l'opérateur

Pour les installations effectuées à l'aide de l'opérateur Trident, le processus de mise à niveau vers une version antérieure est différent et ne nécessite pas l'utilisation de `tridentctl`.

Pour les installations utilisant l'opérateur Trident, il est possible de rétrograder avec l'un des éléments suivants :

- Une version installée à l'aide de l'opérateur à étendue de l'espace de noms (20.07 - 20.10).
- Version installée à l'aide de l'opérateur à étendue de cluster (21.01 et versions ultérieures).

Revenir à l'opérateur du groupe d'instruments

Pour rétrograder d'Astra Trident à une version utilisant le groupe d'opérateurs du cluster, suivez les étapes indiquées ci-dessous.

Étapes

1. "Désinstaller Astra Trident". **Ne supprimez pas les CRD sauf si vous souhaitez supprimer complètement une installation existante.**
2. L'opérateur Trident peut être supprimé via le manifeste de l'opérateur associé à votre version de Trident. Par exemple : `https://github.com/NetApp/trident/tree/stable/vXX.XX /deploy/bundle.yaml` où `vXX.XX` est le numéro de version (par exemple `v22.10`) et `bundle.yaml` Est le nom du fichier YAML du bundle.
3. Poursuivez la rétrogradation en installant la version souhaitée d'Astra Trident. Suivez la documentation de la version souhaitée.

Revenir à l'opérateur de la portée de l'espace de noms

Cette section résume les étapes à suivre pour rétrograder à une version d'Astra Trident comprise entre 20.07 et 20.10, qui sera installée à l'aide de l'opérateur à espace de noms.

Étapes

1. "Désinstaller Astra Trident". **Ne pas utiliser les CRD sauf si vous souhaitez supprimer complètement une installation existante.** Assurez-vous que le `tridentorchestrator` est supprimé.

```
#Check to see if there are any tridentorchestrators present
kubectl get torc
NAME        AGE
trident     20h

#Looks like there is a tridentorchestrator that needs deleting
kubectl delete torc trident
tridentorchestrator.trident.netapp.io "trident" deleted
```

2. L'opérateur Trident peut être supprimé via le manifeste de l'opérateur associé à votre version de Trident. Par exemple : `https://github.com/NetApp/trident/tree/stable/vXX.XX /deploy/bundle.yaml` où `vXX.XX` est le numéro de version (par exemple `v22.10`) et `bundle.yaml` Est le nom du fichier YAML du bundle.
3. Supprimez le `tridentorchestrator` CRD.

```
#Check to see if ``tridentorchestrators.trident.netapp.io`` CRD is
present and delete it.

kubectl get crd tridentorchestrators.trident.netapp.io

NAME                                        CREATED AT
tridentorchestrators.trident.netapp.io   2021-01-21T21:11:37Z

kubectl delete crd tridentorchestrators.trident.netapp.io

customresourcedefinition.apiextensions.k8s.io
"tridentorchestrators.trident.netapp.io" deleted
```

Astra Trident a été désinstallé.

4. Poursuivez la rétrogradation en installant la version souhaitée. Suivez la documentation de la version souhaitée.

Rétrograder en utilisant Helm

Pour rétrograder, utilisez `helm rollback` commande. Voir l'exemple suivant :

```
helm rollback trident [revision #]
```

Processus de rétrogradation lors de l'installation d'Astra Trident à l'aide de `tridentctl`

Si vous avez installé Astra Trident à l'aide de `tridentctl`, le processus de rétrogradation implique les étapes suivantes. Cette séquence vous guide dans le processus de rétrogradation pour passer d'Astra Trident 21.07 à 20.07.

Avant de lancer la restauration vers une version antérieure, vous devez prendre une copie Snapshot de votre cluster Kubernetes `etcd`. Cela vous permet de sauvegarder l'état actuel des CRD d'Astra Trident.

Étapes

1. Assurez-vous que Trident est installé à l'aide de `tridentctl`. Si vous ne savez pas comment Astra Trident est installé, exécutez ce test simple :
   a. Lister les pods présents dans l'espace de noms Trident.
   b. Identifier la version d'Astra Trident exécutée dans votre cluster. Vous pouvez utiliser `tridentctl` Vous pouvez également consulter l'image utilisée dans les pods Trident.
   c. Si vous **ne voyez pas** A. `tridentOrchestrator`, (ou) a `tridentprovisioner`, (ou) un pod nommé `trident-operator-xxxxxxxxxx-xxxxx`, Astra Trident * est installé* avec `tridentctl`.
2. Désinstallez Astra Trident avec l'ancien `tridentctl` binaire. Dans ce cas, vous désinstallerez avec le binaire 21.07.

```
tridentctl version -n trident
+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 21.07.0        | 21.07.0        |
+----------------+----------------+

tridentctl uninstall -n trident
INFO Deleted Trident deployment.
INFO Deleted Trident daemonset.
INFO Deleted Trident service.
INFO Deleted Trident secret.
INFO Deleted cluster role binding.
INFO Deleted cluster role.
INFO Deleted service account.
INFO Deleted pod security policy.
podSecurityPolicy=tridentpods
INFO The uninstaller did not delete Trident's namespace in case it is
going to be reused.
INFO Trident uninstallation succeeded.
```

3. Une fois le fichier terminé, procurez-vous le binaire Trident pour la version souhaitée (dans cet exemple, 20.07) et installez Astra Trident. Vous pouvez générer des YAML personnalisées pour un "installation personnalisée" si nécessaire.

```
cd 20.07/trident-installer/
./tridentctl install -n trident-ns
INFO Created installer service account.
serviceaccount=trident-installer
INFO Created installer cluster role.              clusterrole=trident-
installer
INFO Created installer cluster role binding.
clusterrolebinding=trident-installer
INFO Created installer configmap.                 configmap=trident-
installer
...
...
INFO Deleted installer cluster role binding.
INFO Deleted installer cluster role.
INFO Deleted installer service account.
```

Le processus de rétrogradation est terminé.

Avec Astra Trident

Préparez le nœud de travail

Tous les nœuds workers du cluster Kubernetes doivent pouvoir monter les volumes provisionnés pour vos pods. Pour préparer les nœuds workers, vous devez installer des outils NFS ou iSCSI en fonction de votre sélection de pilotes.

Choisir les bons outils

Si vous utilisez une combinaison de pilotes, vous devez installer les outils NFS et iSCSI.

Outils NFS

Installez les outils NFS si vous utilisez : `ontap-nas`, `ontap-nas-economy`, `ontap-nas-flexgroup`, `azure-netapp-files`, `gcp-cvs`

Outils iSCSI

Installez les outils iSCSI si vous utilisez : `ontap-san`, `ontap-san-economy`, `solidfire-san`

Les versions récentes de RedHat CoreOS ont installé NFS et iSCSI par défaut.

Détection des services de nœud

Astra Trident tente de détecter automatiquement si le nœud peut exécuter des services iSCSI ou NFS.

La découverte de services de nœuds identifie les services détectés, mais ne garantit pas que les services sont correctement configurés. Inversement, l'absence d'un service découvert ne garantit pas l'échec du montage du volume.

Révision des événements

Astra Trident crée des événements pour le nœud afin d'identifier les services découverts. Pour passer en revue ces événements, exécutez :

```
kubectl get event -A --field-selector involvedObject.name=<Kubernetes node name>
```

Examiner les services découverts

Astra Trident identifie les services activés pour chaque nœud du nœud Trident CR. Pour afficher les services découverts, exécutez :

```
tridentctl get node -o wide -n <Trident namespace>
```

Volumes NFS

Installez les outils NFS à l'aide des commandes de votre système d'exploitation. Assurez-vous que le service NFS est démarré pendant le démarrage.

RHEL 8+

```
sudo yum install -y nfs-utils
```

Ubuntu

```
sudo apt-get install -y nfs-common
```

Redémarrez les nœuds workers après l'installation des outils NFS afin d'éviter toute défaillance lors de la connexion des volumes aux conteneurs.

Volumes iSCSI

Astra Trident peut établir automatiquement une session iSCSI, analyser les LUN et détecter les périphériques à chemins d'accès multiples, les formater et les monter sur un pod.

Fonctionnalités d'auto-rétablissement de l'iSCSI

Pour les systèmes ONTAP, Astra Trident exécute l'auto-rétablissement iSCSI toutes les cinq minutes pour :

1. **Identifier** l'état de session iSCSI souhaité et l'état de session iSCSI en cours.
2. **Comparer** l'état souhaité à l'état actuel pour identifier les réparations nécessaires. Astra Trident détermine les priorités de réparation et le moment auquel les réparations doivent être effectuées.
3. **Effectuez les réparations** requises pour rétablir l'état de session iSCSI actuel à l'état de session iSCSI souhaité.

Les journaux d'activité d'auto-rétablissement se trouvent dans le `trident-main` Conteneur sur le pod DemOnset respectif. Pour afficher les journaux, vous devez avoir défini `debug` « Vrai » lors de l'installation d'Astra Trident.

Avec les fonctionnalités d'autorétablissement iSCSI d'Astra Trident,

- Sessions iSCSI obsolètes ou non saines pouvant survenir après un problème de connectivité réseau. Dans le cas d'une session obsolète, Astra Trident attend sept minutes avant de se déconnecter pour rétablir la connexion avec un portail.

  Par exemple, si les secrets CHAP ont tourné sur le contrôleur de stockage et que le réseau perd la connectivité, les anciens secrets CHAP (*obsolète*) pourraient persister. L'auto-rétablissement peut reconnaître ceci et rétablir automatiquement la session pour appliquer les secrets CHAP mis à jour.

- Sessions iSCSI manquantes
- LUN manquantes

Installez les outils iSCSI

Installez les outils iSCSI à l'aide des commandes de votre système d'exploitation.

Avant de commencer

- Chaque nœud du cluster Kubernetes doit avoir un IQN unique. **C'est une condition préalable nécessaire**.
- En cas d'utilisation de RHCOS version 4.5 ou ultérieure ou d'une autre distribution Linux compatible RHEL, avec le `solidfire-san` Pilote et Element OS 12.5 ou version antérieure, assurez-vous que l'algorithme d'authentification CHAP est défini sur MD5 dans `/etc/iscsi/iscsid.conf`. Les algorithmes CHAP sécurisés conformes à la norme FIPS SHA1, SHA-256 et SHA3-256 sont disponibles avec Element 12.7.

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

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

RHEL 8+

1. Installez les packages système suivants :

```
sudo yum install -y lsscsi iscsi-initiator-utils sg3_utils device-
mapper-multipath
```

2. Vérifiez que la version iscsi-initiator-utils est 6.2.0.874-2.el7 ou ultérieure :

```
rpm -q iscsi-initiator-utils
```

3. Définir la numérisation sur manuelle :

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

4. Activer les chemins d'accès multiples :

```
sudo mpathconf --enable --with_multipathd y --find_multipaths n
```

> Bien sûr `etc/multipath.conf` contient `find_multipaths no` sous `defaults`.

5. S'assurer que `iscsid` et `multipathd` sont en cours d'exécution :

```
sudo systemctl enable --now iscsid multipathd
```

6. Activer et démarrer `iscsi`:

```
sudo systemctl enable --now iscsi
```

Ubuntu

1. Installez les packages système suivants :

```
sudo apt-get install -y open-iscsi lsscsi sg3-utils multipath-tools
scsitools
```

2. Vérifiez que la version Open-iscsi est 2.0.874-5ubuntu2.10 ou ultérieure (pour bionique) ou 2.0.874-7.1ubuntu6.1 ou ultérieure (pour focaux) :

```
dpkg -l open-iscsi
```

3. Définir la numérisation sur manuelle :

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

4. Activer les chemins d'accès multiples :

```
sudo tee /etc/multipath.conf <<-'EOF
defaults {
    user_friendly_names yes
    find_multipaths no
}
EOF
sudo systemctl enable --now multipath-tools.service
sudo service multipath-tools restart
```

Bien sûr `etc/multipath.conf` contient `find_multipaths no` sous `defaults`.

5. S'assurer que `open-iscsi` et `multipath-tools` sont activées et en cours d'exécution :

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

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

Redémarrez les nœuds workers après l'installation des outils iSCSI pour éviter toute défaillance lors de la connexion des volumes aux conteneurs.

Configuration des systèmes back-end

Un système back-end définit la relation entre Astra Trident et un système de stockage. Il explique à Astra Trident comment communiquer avec ce système de stockage et comment Astra Trident doit provisionner des volumes à partir de celui-ci.

Astra Trident propose automatiquement des pools de stockage back-end correspondant aux exigences définies par une classe de stockage. Découvrez comment configurer le système back-end pour votre système

de stockage.

- "Configurer un back-end Azure NetApp Files"
- "Configurer un système back-end Cloud Volumes Service pour Google Cloud Platform"
- "Configurer un système NetApp HCI ou SolidFire backend"
- "Configurer un système back-end avec des pilotes NAS ONTAP ou Cloud Volumes ONTAP"
- "Configurer un système back-end avec des pilotes ONTAP ou Cloud Volumes ONTAP SAN"
- "Utilisez Astra Trident avec Amazon FSX pour NetApp ONTAP"

Azure NetApp Files

Configurer un back-end Azure NetApp Files

Vous pouvez configurer Azure NetApp Files (ANF) comme back-end pour Astra Trident. Vous pouvez relier des volumes NFS et SMB à l'aide d'un back-end ANF.

- "Préparation"
- "Exemples et options de configuration"

Considérations

- Le service Azure NetApp Files ne prend pas en charge des volumes de moins de 100 Go. Astra Trident crée automatiquement des volumes de 100 Go en cas de demande d'un volume plus petit.
- Astra Trident prend en charge les volumes SMB montés sur des pods qui s'exécutent uniquement sur des nœuds Windows.
- Astra Trident ne prend pas en charge l'architecture Windows ARM.

Préparez la configuration d'un back-end Azure NetApp Files

Avant de pouvoir configurer le système back-end Azure NetApp Files, vous devez vous assurer que les exigences suivantes sont respectées.

> Si vous utilisez Azure NetApp Files pour la première fois ou dans un nouvel emplacement, une configuration initiale est requise pour configurer Azure NetApp Files et créer un volume NFS. Reportez-vous à la section "Azure : configurez Azure NetApp Files et créez un volume NFS".

Prérequis pour les volumes NFS et SMB

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

- Un pool de capacité. Reportez-vous à la section "Microsoft : créez un pool de capacité pour Azure NetApp Files".
- Sous-réseau délégué à Azure NetApp Files. Reportez-vous à la section "Microsoft : déléguer un sous-réseau à Azure NetApp Files".
- `subscriptionID` Depuis un abonnement Azure avec Azure NetApp Files activé.
- `tenantID`, `clientID`, et `clientSecret` à partir d'un "Enregistrement d'applications" Dans Azure Active Directory avec les autorisations suffisantes pour le service Azure NetApp Files. L'enregistrement de l'application doit utiliser l'une des options suivantes :

- Rôle propriétaire ou contributeur "Prédéfinie par Azure".
- A "Rôle de contributeur personnalisé" au niveau de l'abonnement (`assignableScopes`) Avec les autorisations suivantes qui sont limitées à ce qu'exige Astra Trident. Après avoir créé le rôle personnalisé, "Attribuez le rôle à l'aide du portail Azure".

```
{
    "id": "/subscriptions/<subscription-
id>/providers/Microsoft.Authorization/roleDefinitions/<role-
definition-id>",
    "properties": {
        "roleName": "custom-role-with-limited-perms",
        "description": "custom role providing limited permissions",
        "assignableScopes": [
            "/subscriptions/<subscription-id>"
        ],
        "permissions": [
            {
                "actions": [

"Microsoft.NetApp/netAppAccounts/capacityPools/read",

"Microsoft.NetApp/netAppAccounts/capacityPools/write",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/read",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/write",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/delete",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/snapshots/read
",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/snapshots/writ
e",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/snapshots/dele
te",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/subvolumes/rea
d",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/subvolumes/wri
te",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/subvolumes/del
ete",
```

```
"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/subvolumes/Get
Metadata/action",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/MountTargets/r
ead",
                    "Microsoft.Network/virtualNetworks/read",
                    "Microsoft.Network/virtualNetworks/subnets/read",

"Microsoft.Features/featureProviders/subscriptionFeatureRegistrations
/read",

"Microsoft.Features/featureProviders/subscriptionFeatureRegistrations
/write",

"Microsoft.Features/featureProviders/subscriptionFeatureRegistrations
/delete",
                    "Microsoft.Features/features/read",
                    "Microsoft.Features/operations/read",
                    "Microsoft.Features/providers/features/read",

"Microsoft.Features/providers/features/register/action",

"Microsoft.Features/providers/features/unregister/action",

"Microsoft.Features/subscriptionFeatureRegistrations/read"
                ],
                "notActions": [],
                "dataActions": [],
                "notDataActions": []
            }
        ]
    }
}
```

- Azure `location` qui contient au moins un "sous-réseau délégué". À partir de Trident 22.01, le `location` le paramètre est un champ obligatoire au niveau supérieur du fichier de configuration back-end. Les valeurs d'emplacement spécifiées dans les pools virtuels sont ignorées.

Exigences supplémentaires pour les volumes SMB

Pour créer un volume SMB, vous devez disposer des éléments suivants :

- Active Directory configuré et connecté à Azure NetApp Files. Reportez-vous à la section "Microsoft : création et gestion des connexions Active Directory pour Azure NetApp Files".
- Cluster Kubernetes avec un nœud de contrôleur Linux et au moins un nœud worker Windows exécutant Windows Server 2019. Astra Trident prend en charge les volumes SMB montés sur des pods qui

s'exécutent uniquement sur des nœuds Windows.

- Au moins un secret Astra Trident contenant vos informations d'identification Active Directory pour que Azure NetApp Files puisse s'authentifier auprès d'Active Directory. Pour générer un secret `smbcreds`:

```
kubectl create secret generic smbcreds --from-literal username=user
--from-literal password='password'
```

- Un proxy CSI configuré en tant que service Windows. Pour configurer un `csi-proxy`, voir "GitHub : proxy CSI" ou "GitHub : proxy CSI pour Windows" Pour les nœuds Kubernetes s'exécutant sur Windows.

Exemples et options de configuration du back-end Azure NetApp Files

Découvrez les options de configuration du back-end NFS et SMB pour ANF et examinez les exemples de configuration.

Astra Trident utilise votre configuration back-end (sous-réseau, réseau virtuel, niveau de service et emplacement), pour créer des volumes ANF dans des pools de capacité disponibles à l'emplacement demandé et correspondre au niveau de service et au sous-réseau requis.

Astra Trident ne prend pas en charge les pools de capacité manuels de QoS.

Options de configuration du back-end

Les systèmes back-end ANF proposent ces options de configuration.

Paramètre	Description	Valeur par défaut
`version`		Toujours 1
`storageDriverName`	Nom du pilote de stockage	« azure-netapp-files »
`backendName`	Nom personnalisé ou système back-end de stockage	Nom du pilote + "_" + caractères aléatoires
`subscriptionID`	L'ID d'abonnement de votre abonnement Azure	
`tenantID`	ID locataire d'un enregistrement d'application	
`clientID`	L'ID client d'un enregistrement d'application	
`clientSecret`	Secret client d'un enregistrement d'application	
`serviceLevel`	Un de `Standard`, `Premium`, ou `Ultra`	« » (aléatoire)
`location`	Nom de l'emplacement Azure dans lequel les nouveaux volumes seront créés	

Paramètre	Description	Valeur par défaut
`resourceGroups`	Liste des groupes de ressources pour le filtrage des ressources découvertes	«[] » (sans filtre)
`netappAccounts`	Liste des comptes NetApp permettant de filtrer les ressources découvertes	«[] » (sans filtre)
`capacityPools`	Liste des pools de capacité pour le filtrage des ressources découvertes	«[] » (sans filtre, aléatoire)
`virtualNetwork`	Nom d'un réseau virtuel avec un sous-réseau délégué	« »
`subnet`	Nom d'un sous-réseau délégué à `Microsoft.Netapp/volumes`	« »
`networkFeatures`	L'ensemble des fonctions de vnet pour un volume peut être `Basic` ou `Standard`. Les fonctions réseau ne sont pas disponibles dans toutes les régions et peuvent être activées dans un abonnement. Spécification `networkFeatures` lorsque la fonctionnalité n'est pas activée, le provisionnement du volume échoue.	« »
`nfsMountOptions`	Contrôle précis des options de montage NFS. Ignoré pour les volumes SMB. Pour monter des volumes à l'aide de NFS version 4.1, incluez `nfsvers=4` Dans la liste des options de montage délimitées par des virgules, choisissez NFS v4.1. Les options de montage définies dans une définition de classe de stockage remplacent les options de montage définies dans la configuration backend.	« nfsvers=3 »
`limitVolumeSize`	Echec du provisionnement si la taille du volume demandé est supérieure à cette valeur	« » (non appliqué par défaut)
`debugTraceFlags`	Indicateurs de débogage à utiliser lors du dépannage. Exemple `\{"api": false, "method": true, "discovery": true}`. Ne l'utilisez pas à moins que vous ne soyez en mesure de résoudre les problèmes et que vous ayez besoin d'un vidage détaillé des journaux.	nul

Paramètre	Description	Valeur par défaut
`nasType`	Configurez la création de volumes NFS ou SMB. Les options sont `nfs`, `smb` ou nul. La valeur null par défaut sur les volumes NFS.	`nfs`

Pour plus d'informations sur les fonctionnalités réseau, reportez-vous à la section "Configurer les fonctions réseau d'un volume Azure NetApp Files".

Autorisations et ressources requises

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

Les valeurs de `resourceGroups`, `netappAccounts`, `capacityPools`, `virtualNetwork`, et `subnet` peut être spécifié à l'aide de noms courts ou complets. Les noms complets sont recommandés dans la plupart des cas, car les noms abrégés peuvent faire correspondre plusieurs ressources avec le même nom.

Le `resourceGroups`, `netappAccounts`, et `capacityPools` les valeurs sont des filtres qui limitent l'ensemble des ressources découvertes aux ressources disponibles pour ce stockage back-end et peuvent être spécifiés dans n'importe quelle combinaison. Les noms complets suivent le format suivant :

Type	Format
Groupe de ressources	<groupe de ressources>
Compte NetApp	<groupe de ressources>/<compte netapp>
Pool de capacité	<groupe de ressources>/<compte netapp>/<pool de capacité>
Réseau virtuel	<groupe de ressources>/<réseau virtuel>
Sous-réseau	<groupe de ressources>/<réseau virtuel>/<sous-réseau>

Provisionnement de volume

Vous pouvez contrôler le provisionnement de volume par défaut en spécifiant les options suivantes dans une section spéciale du fichier de configuration. Reportez-vous à la section Exemples de configurations pour plus d'informations.

Paramètre	Description	Valeur par défaut
`exportRule`	Règles d'exportation pour les nouveaux volumes. `exportRule` Doit être une liste séparée par des virgules d'une combinaison d'adresses IPv4 ou de sous-réseaux IPv4 en notation CIDR. Ignoré pour les volumes SMB.	« 0.0.0.0/0 »
`snapshotDir`	Contrôle la visibilité du répertoire .snapshot	« faux »
`size`	Taille par défaut des nouveaux volumes	« 100 G »
`unixPermissions`	Les autorisations unix des nouveaux volumes (4 chiffres octaux). Ignoré pour les volumes SMB.	« » (fonction d'aperçu, liste blanche requise dans l'abonnement)

Exemples de configurations

Exemple 1 : configuration minimale

Il s'agit de la configuration back-end minimale absolue. Avec cette configuration, Astra Trident détecte tous vos comptes, pools de capacité et sous-réseaux NetApp délégués à ANF à l'emplacement configuré et place les nouveaux volumes sur l'un de ces pools et sous-réseaux de manière aléatoire. Parce que `nasType` est omis, le `nfs` La valeur par défaut s'applique et le système back-end provisionne les volumes NFS.

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

```
---
version: 1
storageDriverName: azure-netapp-files
subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
tenantID: 68e4f836-edc1-fake-bff9-b2d865ee56cf
clientID: dd043f63-bf8e-fake-8076-8de91e5713aa
clientSecret: SECRET
location: eastus
```

Exemple 2 : configuration de niveau de service spécifique avec des filtres de pool de capacité

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

```
---
version: 1
storageDriverName: azure-netapp-files
subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
tenantID: 68e4f836-edc1-fake-bff9-b2d865ee56cf
clientID: dd043f63-bf8e-fake-8076-8de91e5713aa
clientSecret: SECRET
location: eastus
serviceLevel: Ultra
capacityPools:
- application-group-1/account-1/ultra-1
- application-group-1/account-1/ultra-2
```

Exemple 3 : configuration avancée

Cette configuration back-end réduit davantage l'étendue du placement des volumes sur un seul sous-réseau et modifie également certains paramètres par défaut du provisionnement des volumes.

```
---
version: 1
storageDriverName: azure-netapp-files
subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
tenantID: 68e4f836-edc1-fake-bff9-b2d865ee56cf
clientID: dd043f63-bf8e-fake-8076-8de91e5713aa
clientSecret: SECRET
location: eastus
serviceLevel: Ultra
capacityPools:
- application-group-1/account-1/ultra-1
- application-group-1/account-1/ultra-2
virtualNetwork: my-virtual-network
subnet: my-subnet
networkFeatures: Standard
nfsMountOptions: vers=3,proto=tcp,timeo=600
limitVolumeSize: 500Gi
defaults:
  exportRule: 10.0.0.0/24,10.0.1.0/24,10.0.2.100
  snapshotDir: 'true'
  size: 200Gi
  unixPermissions: '0777'
```

Exemple 4 : configuration de pool virtuel

Cette configuration back-end définit plusieurs pools de stockage dans un seul fichier. Cette fonction est utile lorsque plusieurs pools de capacité prennent en charge différents niveaux de service, et que vous souhaitez créer des classes de stockage dans Kubernetes qui les représentent. Des étiquettes de pools virtuels ont été utilisées pour différencier les pools en fonction de `performance`.

```
---
version: 1
storageDriverName: azure-netapp-files
subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
tenantID: 68e4f836-edc1-fake-bff9-b2d865ee56cf
clientID: dd043f63-bf8e-fake-8076-8de91e5713aa
clientSecret: SECRET
location: eastus
resourceGroups:
- application-group-1
networkFeatures: Basic
nfsMountOptions: vers=3,proto=tcp,timeo=600
labels:
  cloud: azure
storage:
- labels:
    performance: gold
  serviceLevel: Ultra
  capacityPools:
  - ultra-1
  - ultra-2
  networkFeatures: Standard
- labels:
    performance: silver
  serviceLevel: Premium
  capacityPools:
  - premium-1
- labels:
    performance: bronze
  serviceLevel: Standard
  capacityPools:
  - standard-1
  - standard-2
```

Définitions des classes de stockage

Les éléments suivants `StorageClass` les définitions font référence aux pools de stockage ci-dessus.

Exemples de définitions utilisant `parameter.selector` légale

À l'aide de `parameter.selector` vous pouvez spécifier pour chaque `StorageClass` pool virtuel utilisé pour héberger un volume. Les aspects définis dans le pool sélectionné seront définis pour le volume.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gold
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=gold"
allowVolumeExpansion: true
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: silver
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=silver"
allowVolumeExpansion: true
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: bronze
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=bronze"
allowVolumeExpansion: true
```

Exemples de définitions pour les volumes SMB

À l'aide de `nasType`, `node-stage-secret-name`, et `node-stage-secret-namespace`, Vous pouvez spécifier un volume SMB et fournir les informations d'identification Active Directory requises.

Exemple 1 : configuration de base sur l'espace de noms par défaut

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: anf-sc-smb
provisioner: csi.trident.netapp.io
parameters:
  backendType: "azure-netapp-files"
  trident.netapp.io/nasType: "smb"
  csi.storage.k8s.io/node-stage-secret-name: "smbcreds"
  csi.storage.k8s.io/node-stage-secret-namespace: "default"
```

Exemple 2 : utilisation de secrets différents par espace de noms

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: anf-sc-smb
provisioner: csi.trident.netapp.io
parameters:
  backendType: "azure-netapp-files"
  trident.netapp.io/nasType: "smb"
  csi.storage.k8s.io/node-stage-secret-name: "smbcreds"
  csi.storage.k8s.io/node-stage-secret-namespace: ${pvc.namespace}
```

Exemple 3 : utilisation de différents secrets par volume

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: anf-sc-smb
provisioner: csi.trident.netapp.io
parameters:
  backendType: "azure-netapp-files"
  trident.netapp.io/nasType: "smb"
  csi.storage.k8s.io/node-stage-secret-name: ${pvc.name}
  csi.storage.k8s.io/node-stage-secret-namespace: ${pvc.namespace}
```

`nasType: `smb` Filtres pour les pools qui prennent en charge les volumes SMB. `nasType: `nfs` ou `nasType: `null` Filtres pour pools NFS.

Créer le backend

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

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

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

```
tridentctl logs
```

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

Configurer un système Cloud Volumes Service pour Google Cloud backend

Découvrez comment configurer NetApp Cloud Volumes Service pour Google Cloud en tant que backend pour votre installation d'Astra Trident à l'aide des exemples de configuration fournis.

En savoir plus sur la prise en charge d'Astra Trident pour Cloud Volumes Service pour Google Cloud

Astra Trident peut créer des volumes Cloud Volumes Service dans un des deux "types de service":

- **CVS-Performance** : le type de service Astra Trident par défaut. Ce type de service aux performances optimisées est parfaitement adapté aux charges de travail de production qui exigent des performances élevées. Le type de service CVS-Performance est une option matérielle prenant en charge les volumes d'une taille minimale de 100 Gio. Vous pouvez choisir l'une des options "trois niveaux de service":
  - `standard`
  - `premium`
  - `extreme`
- **CVS**: Le type de service CVS fournit une haute disponibilité zonale avec des niveaux de performance limités à modérés. Le type de service CVS est une option logicielle utilisant des pools de stockage pour prendre en charge des volumes de 1 Gio. Le pool de stockage peut contenir jusqu'à 50 volumes dans lesquels tous les volumes partagent la capacité et les performances du pool. Vous pouvez choisir l'une des options "deux niveaux de service":
  - `standardsw`
  - `zoneredundantstandardsw`

Ce dont vous avez besoin

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

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

Options de configuration du back-end

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

Paramètre	Description	Valeur par défaut
`version`		Toujours 1
`storageDriverName`	Nom du pilote de stockage	« gcp-cvs »
`backendName`	Nom personnalisé ou système back-end de stockage	Nom du pilote + "_" + partie de la clé API
`storageClass`	Paramètre facultatif utilisé pour spécifier le type de service CVS. Utiliser `software` Pour sélectionner le type de service CVS. Sinon, Astra Trident suppose un type de service CVS-Performance (`hardware`).	
`storagePools`	Type de service CVS uniquement. Paramètre facultatif utilisé pour spécifier les pools de stockage pour la création du volume.	
`projectNumber`	Numéro de projet de compte Google Cloud. La valeur est disponible sur la page d'accueil du portail Google Cloud.	
`hostProjectNumber`	Requis si l'utilisation d'un réseau VPC partagé. Dans ce scénario, `projectNumber` est le projet de service, et `hostProjectNumber` est le projet hôte.	
`apiRegion`	Région Google Cloud dans laquelle Astra Trident crée des volumes Cloud Volumes Service. Lors de la création de clusters Kubernetes inter-région, de volumes créés dans un `apiRegion` Peut être utilisé pour des charges de travail planifiées sur des nœuds sur plusieurs régions Google Cloud. Le trafic entre les régions coûte plus cher.	

Paramètre	Description	Valeur par défaut
`apiKey`	Clé API pour le compte de service Google Cloud avec le `netappcloudvolumes.admin` rôle. Il inclut le contenu au format JSON du fichier de clé privée d'un compte de service Google Cloud (copié en compte dans le fichier de configuration back-end).	
`proxyURL`	URL proxy si le serveur proxy doit se connecter au compte CVS. Le serveur proxy peut être un proxy HTTP ou HTTPS. Pour un proxy HTTPS, la validation du certificat est ignorée pour permettre l'utilisation de certificats auto-signés dans le serveur proxy. Les serveurs proxy avec authentification activée ne sont pas pris en charge.	
`nfsMountOptions`	Contrôle précis des options de montage NFS.	« nfsvers=3 »
`limitVolumeSize`	Echec du provisionnement si la taille du volume demandé est supérieure à cette valeur.	« » (non appliqué par défaut)
`serviceLevel`	Niveau de service CVS-Performance ou CVS pour les nouveaux volumes. Les valeurs CVS-Performance sont `standard`, `premium`, ou `extreme`. Les valeurs CVS sont `standardsw` ou `zoneredundantstandardsw`.	CVS-Performance par défaut est « standard ». CVS default est "standardsw".
`network`	Réseau Google Cloud utilisé pour les volumes Cloud Volumes Service.	« par défaut »
`debugTraceFlags`	Indicateurs de débogage à utiliser lors du dépannage. Exemple `\{"api":false, "method":true}`. Ne l'utilisez pas à moins que vous ne soyez en mesure de résoudre les problèmes et que vous ayez besoin d'un vidage détaillé des journaux.	nul

Paramètre	Description	Valeur par défaut

`allowedTopologies`	Pour activer l'accès inter-région, votre définition de classe de stockage pour `allowedTopologies` doit inclure toutes les régions. Par exemple :
	`- key: topology.kubernetes.io/region`
	`values:`
	`- us-east1`
	`- europe-west1`	

Options de provisionnement de volumes

Vous pouvez contrôler le provisionnement de volume par défaut dans le `defaults` section du fichier de configuration.

Paramètre	Description	Valeur par défaut
`exportRule`	Règles d'exportation pour les nouveaux volumes. Doit être une liste séparée par des virgules d'une combinaison d'adresses IPv4 ou de sous-réseaux IPv4 en notation CIDR.	« 0.0.0.0/0 »
`snapshotDir`	Accès au `.snapshot` répertoire	« faux »
`snapshotReserve`	Pourcentage de volume réservé pour les snapshots	« » (Accepter CVS par défaut de 0)
`size`	La taille des nouveaux volumes. CVS-Performance minimum est de 100 Gio. CVS est au minimum de 1 Gio.	Le type de service CVS-Performance utilise par défaut « 100 Gio ». Le type de service CVS n'est pas défini par défaut mais nécessite au moins 1 Gio.

Exemples de type de service CVS-Performance

Les exemples suivants fournissent des exemples de configuration pour le type de service CVS-Performance.

Exemple 1 : configuration minimale

Il s'agit de la configuration back-end minimale avec le type de service CVS-Performance par défaut et le niveau de service « standard » par défaut.

```
---
version: 1
storageDriverName: gcp-cvs
projectNumber: '012345678901'
apiRegion: us-west2
apiKey:
  type: service_account
  project_id: my-gcp-project
  private_key_id: "<id_value>"
  private_key: |
    -----BEGIN PRIVATE KEY-----
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    XsYg6gyxy4zq7OlwWgLwGa==
    -----END PRIVATE KEY-----
  client_email: cloudvolumes-admin-sa@my-gcp-
project.iam.gserviceaccount.com
  client_id: '123456789012345678901'
```

```
  auth_uri: https://accounts.google.com/o/oauth2/auth
  token_uri: https://oauth2.googleapis.com/token
  auth_provider_x509_cert_url:
https://www.googleapis.com/oauth2/v1/certs
  client_x509_cert_url:
https://www.googleapis.com/robot/v1/metadata/x509/cloudvolumes-admin-
sa%40my-gcp-project.iam.gserviceaccount.com
```

Exemple 2 : configuration du niveau de service

Dans cet exemple, nous présentons les options de configuration du back-end, y compris les niveaux de service et les valeurs par défaut des volumes.

```
---
version: 1
storageDriverName: gcp-cvs
projectNumber: '012345678901'
apiRegion: us-west2
apiKey:
  type: service_account
  project_id: my-gcp-project
  private_key_id: "<id_value>"
  private_key: |
    -----BEGIN PRIVATE KEY-----
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    XsYg6gyxy4zq7OlwWgLwGa==
    -----END PRIVATE KEY-----
  client_email: cloudvolumes-admin-sa@my-gcp-
project.iam.gserviceaccount.com
  client_id: '123456789012345678901'
```

```
  auth_uri: https://accounts.google.com/o/oauth2/auth
  token_uri: https://oauth2.googleapis.com/token
  auth_provider_x509_cert_url:
https://www.googleapis.com/oauth2/v1/certs
  client_x509_cert_url:
https://www.googleapis.com/robot/v1/metadata/x509/cloudvolumes-admin-
sa%40my-gcp-project.iam.gserviceaccount.com
proxyURL: http://proxy-server-hostname/
nfsMountOptions: vers=3,proto=tcp,timeo=600
limitVolumeSize: 10Ti
serviceLevel: premium
defaults:
  snapshotDir: 'true'
  snapshotReserve: '5'
  exportRule: 10.0.0.0/24,10.0.1.0/24,10.0.2.100
  size: 5Ti
```

Exemple 3 : configuration de pool virtuel

Utilisation de cet échantillon `storage` pour configurer des pools virtuels et `StorageClasses` cela leur renvoie. Reportez-vous à la section Définitions des classes de stockage pour voir comment les classes de stockage ont été définies.

Ici, des valeurs par défaut spécifiques sont définies pour tous les pools virtuels, qui définissent le `snapshotReserve` à 5 % et le `exportRule` à 0.0.0.0/0. Les pools virtuels sont définis dans le `storage` section. Chaque pool virtuel individuel définit sa propre définition `serviceLevel`, et certains pools remplacent les valeurs par défaut. Des étiquettes de pools virtuels ont été utilisées pour différencier les pools en fonction de `performance` et `protection`.

```
---
version: 1
storageDriverName: gcp-cvs
projectNumber: '012345678901'
apiRegion: us-west2
apiKey:
  type: service_account
  project_id: my-gcp-project
  private_key_id: "<id_value>"
  private_key: |
    -----BEGIN PRIVATE KEY-----
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
```

```
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    XsYg6gyxy4zq7OlwWgLwGa==
    -----END PRIVATE KEY-----
  client_email: cloudvolumes-admin-sa@my-gcp-
project.iam.gserviceaccount.com
  client_id: '123456789012345678901'
  auth_uri: https://accounts.google.com/o/oauth2/auth
  token_uri: https://oauth2.googleapis.com/token
  auth_provider_x509_cert_url:
https://www.googleapis.com/oauth2/v1/certs
  client_x509_cert_url:
https://www.googleapis.com/robot/v1/metadata/x509/cloudvolumes-admin-
sa%40my-gcp-project.iam.gserviceaccount.com
nfsMountOptions: vers=3,proto=tcp,timeo=600
defaults:
  snapshotReserve: '5'
  exportRule: 0.0.0.0/0
labels:
  cloud: gcp
region: us-west2
storage:
- labels:
    performance: extreme
    protection: extra
  serviceLevel: extreme
  defaults:
    snapshotDir: 'true'
    snapshotReserve: '10'
    exportRule: 10.0.0.0/24
- labels:
    performance: extreme
    protection: standard
  serviceLevel: extreme
- labels:
    performance: premium
    protection: extra
  serviceLevel: premium
  defaults:
    snapshotDir: 'true'
    snapshotReserve: '10'
- labels:
    performance: premium
    protection: standard
  serviceLevel: premium
- labels:
    performance: standard
```

```
    serviceLevel: standard
```

Définitions des classes de stockage

Les définitions de classe de stockage suivantes s'appliquent à l'exemple de configuration de pool virtuel. À l'aide de `parameters.selector`, Vous pouvez spécifier pour chaque classe de stockage le pool virtuel utilisé pour héberger un volume. Les aspects définis dans le pool sélectionné seront définis pour le volume.

Exemple de classe de stockage

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

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

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

Exemples de type de service CVS

Les exemples suivants fournissent des exemples de configuration pour le type de service CVS.

Exemple 1 : configuration minimale

Il s'agit de la configuration back-end minimale utilisant `storageClass` Pour spécifier le type de service CVS et la valeur par défaut `standardsw` niveau de service.

```
---
version: 1
storageDriverName: gcp-cvs
projectNumber: '012345678901'
storageClass: software
apiRegion: us-east4
apiKey:
  type: service_account
  project_id: my-gcp-project
  private_key_id: "<id_value>"
  private_key: |
    -----BEGIN PRIVATE KEY-----
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    XsYg6gyxy4zq7OlwWgLwGa==
    -----END PRIVATE KEY-----
  client_email: cloudvolumes-admin-sa@my-gcp-
project.iam.gserviceaccount.com
```

```
  client_id: '123456789012345678901'
  auth_uri: https://accounts.google.com/o/oauth2/auth
  token_uri: https://oauth2.googleapis.com/token
  auth_provider_x509_cert_url:
https://www.googleapis.com/oauth2/v1/certs
  client_x509_cert_url:
https://www.googleapis.com/robot/v1/metadata/x509/cloudvolumes-admin-
sa%40my-gcp-project.iam.gserviceaccount.com
serviceLevel: standardsw
```

Exemple 2 : configuration du pool de stockage

Cet exemple de configuration back-end utilise `storagePools` pour configurer un pool de stockage.

```
---
version: 1
storageDriverName: gcp-cvs
backendName: gcp-std-so-with-pool
projectNumber: '531265380079'
apiRegion: europe-west1
apiKey:
  type: service_account
  project_id: cloud-native-data
  private_key_id: "<id_value>"
  private_key: |-
    -----BEGIN PRIVATE KEY-----
    MIIEvAIBADANBgkqhkiG9w0BAQEFAASCBKYwggSiAgEAAoIBAQDaT+Oui9FBAw19
    L1AGEkrYU5xd9K5NlO5jMkIFND5wCD+Nv+jd1GvtFRLaLK5RvXyF5wzvztmODNS+
    qtScpQ+5cFpQkuGtv9U9+N6qtuVYYO3b504Kp5CtqVPJCgMJaK2j8pZTIqUiMum/
    5/Y9oTbZrjAHSMgJm2nHzFq2X0rqVMaHghI6ATm4DOuWx8XGWKTGIPlc0qPqJlqS
    LLaWOH4VIZQZCAyW5IUp9CAmwqHgdG0uhFNfCgMmED6PBUvVLsLvcq86X+QSWR9k
    ETqElj/sGCenPF7ti1DhGBFafd9hPnxg9PZY29ArEZwY9G/ZjZQX7WPgs0VvxiNR
    DxZRC3GXAgMBAAECggEACn5c59bG/qnVEVI1CwMAalM5M2z09JFhlLlljKwntNPj
    Vilw2eTW2+UE7HbJru/S7KQgA5Dnn9kvCraEahPRuddUMrD0vG4kTl/IODV6uFuk
    Y0sZfbqd4jMUQ21smvGsqFzwloYWS5qzO1W83ivXH/HW/iqkmY2eW+EPRS/hwSSu
    SscR+SojI7PB0BWSJhlV4yqYf3vcD/D95el2CVHfRCkL85DKumeZ+yHEnpiXGZAE
    t8xSs4a5OOPm6NHhevCw2a/UQ95/foXNUR450HtbjieJo5o+FF6EYZQGfU2ZHZO8
    37FBKuaJkdGW5xqaI9TL7aqkGkFMF4F2qvOZM+vy8QKBgQD4oVuOkJDlhkTHP86W
    esFlw1kpWyJR9ZA7LI0g/rVpslnX+XdDq0WQf4umdLNau5hYEH9LU6ZSGs1Xk3/B
    NHwR6OXFuqEKNiu83d0zSlHhTy7PZpOZdj5a/vVvQfPDMz7OvsqLRd7YCAbdzuQ0
    +Ahq0Ztwvg0HQ64hdW0ukpYRRwKBgQDgyHj98oqswoYuIa+pP1yS0pPwLmjwKyNm
    /HayzCp+Qjiyy7Tzg8AUqlH1Ou83XbV428jvg7kDhO7PCCKFq+mMmfqHmTpb0Maq
    KpKnZg4ipsqP1yHNNEoRmcailXbwIhCLewMqMrggUiLOmCw4PscL5nK+4GKu2XE1
    jLqjWAZFMQKBgFHkQ9XXRAJ1kR3XpGHoGN890pZOkCVSrqju6aUef/5KYlFCt8ew
    F/+aIxM2iQSvmWQYOvVCnhuY/F2GFaQ7d0om3decuwI0CX/xy7PjHMkLXa2uaZs4
    WR17sLduj62RqXRLX0c0QkwBiNFyHbRcpdkZJQujbYMhBa+7j7SxT4BtAoGAWMWT
    UucocRXZm/pdvz9wteNH3YDWnJLMxm1KC06qMXbBoYrliY4sm3ywJWMC+iCd/H8A
    Gecxd/xVu5mA2L2N3KMq18Zhz8Th0G5DwKyDRJgOQ0Q46yuNXOoYEjlo4Wjyk8Me
    +tlQ8iK98E0UmZnhTgfSpSNElbz2AqnzQ3MN9uECgYAqdvdVPnKGfvdtZ2DjyMoJ
    E89UIC41WjjJGmHsd8W65+3X0RwMzKMT6aZc5tK9J5dHvmWIETnbM+lTImdBBFga
    NWOC6f3r2xbGXHhaWSl+nobpTuvlo56ZRJVvVk7lFMsiddzMuHH8pxfgNJemwA4P
    ThDHCejv035NNV6KyoO0tA==
    -----END PRIVATE KEY-----
  client_email: cloudvolumes-admin-sa@cloud-native-
data.iam.gserviceaccount.com
  client_id: '107071413297115343396'
```

```
  auth_uri: https://accounts.google.com/o/oauth2/auth
  token_uri: https://oauth2.googleapis.com/token
  auth_provider_x509_cert_url:
https://www.googleapis.com/oauth2/v1/certs
  client_x509_cert_url:
https://www.googleapis.com/robot/v1/metadata/x509/cloudvolumes-admin-
sa%40cloud-native-data.iam.gserviceaccount.com
storageClass: software
zone: europe-west1-b
network: default
storagePools:
- 1bc7f380-3314-6005-45e9-c7dc8c2d7509
serviceLevel: Standardsw
```

Et la suite ?

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

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

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

```
tridentctl logs
```

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

Configurer un système NetApp HCI ou SolidFire backend

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

Ce dont vous avez besoin

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

Ce que vous devez savoir

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

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

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

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

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

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

Options de configuration du back-end

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

Paramètre	Description	Valeur par défaut
`version`		Toujours 1
`storageDriverName`	Nom du pilote de stockage	Toujours « solidfire-san ».
`backendName`	Nom personnalisé ou système back-end de stockage	"SolidFire_" + adresse IP de stockage (iSCSI)
`Endpoint`	MVIP pour le cluster SolidFire avec les identifiants de locataire	
`SVIP`	Port et adresse IP de stockage (iSCSI)	

Paramètre	Description	Valeur par défaut
`labels`	Ensemble d'étiquettes arbitraires au format JSON à appliquer aux volumes.	« »
`TenantName`	Nom du locataire à utiliser (créé si introuvable)	
`InitiatorIFace`	Limitez le trafic iSCSI à une interface hôte spécifique	« par défaut »
`UseCHAP`	Utilisez CHAP pour authentifier iSCSI	vrai
`AccessGroups`	Liste des ID de groupes d'accès à utiliser	Recherche l'ID d'un groupe d'accès nommé « trident »
`Types`	Spécifications de QoS	
`limitVolumeSize`	Echec du provisionnement si la taille du volume demandé est supérieure à cette valeur	« » (non appliqué par défaut)
`debugTraceFlags`	Indicateurs de débogage à utiliser lors du dépannage. Exemple, {"api":false, "méthode":true}	nul

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

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

Cet exemple montre un fichier back-end utilisant l'authentification CHAP et la modélisation de trois types de volumes avec des garanties de QoS spécifiques. Il est fort probable que vous définiriez ensuite des classes de stockage pour consommer chacune de ces catégories à l'aide de l' `IOPS` paramètre de classe de stockage.

```
---
version: 1
storageDriverName: solidfire-san
Endpoint: https://<user>:<password>@<mvip>/json-rpc/8.0
SVIP: "<svip>:3260"
TenantName: "<tenant>"
labels:
  k8scluster: dev1
  backend: dev1-element-cluster
UseCHAP: true
Types:
- Type: Bronze
  Qos:
    minIOPS: 1000
    maxIOPS: 2000
    burstIOPS: 4000
- Type: Silver
  Qos:
    minIOPS: 4000
    maxIOPS: 6000
    burstIOPS: 8000
- Type: Gold
  Qos:
    minIOPS: 6000
    maxIOPS: 8000
    burstIOPS: 10000
```

Exemple 2 : configuration du back-end et de la classe de stockage pour `solidfire-san` pilote avec pools virtuels

Cet exemple représente le fichier de définition du back-end configuré avec des pools virtuels ainsi que des classes de stockage qui les renvoient.

Astra Trident copie les étiquettes présentes sur un pool de stockage vers le LUN de stockage back-end lors du provisionnement. Pour plus de commodité, les administrateurs du stockage peuvent définir des étiquettes par pool virtuel et les volumes de groupe par étiquette.

Dans l'exemple de fichier de définition de back-end illustré ci-dessous, des valeurs par défaut spécifiques sont définies pour tous les pools de stockage, qui définissent le `type` Du niveau Silver. Les pools virtuels sont définis dans le `storage` section. Dans cet exemple, certains pools de stockage définissent leur propre type et certains pools remplacent les valeurs par défaut définies ci-dessus.

```
---
version: 1
storageDriverName: solidfire-san
Endpoint: https://<user>:<password>@<mvip>/json-rpc/8.0
```

```
SVIP: "<svip>:3260"
TenantName: "<tenant>"
UseCHAP: true
Types:
- Type: Bronze
  Qos:
    minIOPS: 1000
    maxIOPS: 2000
    burstIOPS: 4000
- Type: Silver
  Qos:
    minIOPS: 4000
    maxIOPS: 6000
    burstIOPS: 8000
- Type: Gold
  Qos:
    minIOPS: 6000
    maxIOPS: 8000
    burstIOPS: 10000
type: Silver
labels:
  store: solidfire
  k8scluster: dev-1-cluster
region: us-east-1
storage:
- labels:
    performance: gold
    cost: '4'
  zone: us-east-1a
  type: Gold
- labels:
    performance: silver
    cost: '3'
  zone: us-east-1b
  type: Silver
- labels:
    performance: bronze
    cost: '2'
  zone: us-east-1c
  type: Bronze
- labels:
    performance: silver
    cost: '1'
  zone: us-east-1d
```

Les définitions de classe de stockage suivantes font référence aux pools virtuels ci-dessus. À l'aide du

`parameters.selector` Chaque classe de stockage indique quel(s) pool(s) virtuel(s) peut(s) être utilisé(s) pour héberger un volume. Les aspects définis dans le pool virtuel sélectionné seront définis pour le volume.

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

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: solidfire-gold-four
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=gold; cost=4"
  fsType: "ext4"
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: solidfire-silver-three
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=silver; cost=3"
  fsType: "ext4"
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: solidfire-bronze-two
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=bronze; cost=2"
  fsType: "ext4"
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: solidfire-silver-one
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=silver; cost=1"
  fsType: "ext4"
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: solidfire-silver
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=silver"
  fsType: "ext4"
```

Trouvez plus d'informations

- "Groupes d'accès de volume"

Configurer un système back-end avec les pilotes SAN ONTAP

Découvrez comment configurer un back-end ONTAP avec les pilotes ONTAP et Cloud Volumes ONTAP SAN.

- "Préparation"
- "Configuration et exemples"

> Astra Control assure une protection, une reprise d'activité et une mobilité transparentes (en déplaçant des volumes entre les clusters Kubernetes) pour les volumes créés avec le système `ontap-nas`, `ontap-nas-flexgroup`, et `ontap-san` pilotes. Voir "Conditions préalables à la réplication d'Astra Control" pour plus d'informations.
>
> - Vous devez utiliser `ontap-nas` adapté aux charges de travail de production qui nécessitent une protection des données, une reprise d'activité et la mobilité.
> - Utiliser `ontap-san-economy` Lorsque vous prévoyez une utilisation de volume, celle-ci devrait être bien supérieure à celle prise en charge par ONTAP.
> - Utiliser `ontap-nas-economy` Ce n'est que lorsque l'utilisation prévue des volumes sera beaucoup plus élevée que ce que prend en charge ONTAP, et le `ontap-san-economy` le pilote ne peut pas être utilisé.
> - Ne pas utiliser `ontap-nas-economy` si vous prévoyez d'avoir besoin en termes de protection des données, de reprise sur incident ou de mobilité.

Autorisations utilisateur

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

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

S'il est possible de créer un rôle plus restrictif au sein de ONTAP qu'un pilote Trident peut utiliser, nous ne le recommandons pas. La plupart des nouvelles versions de Trident appellent des API supplémentaires qui devront être prises en compte, ce qui complique les mises à niveau et risque d'erreurs.

Préparez la configuration du système back-end avec les pilotes SAN ONTAP

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

N'oubliez pas que vous pouvez également exécuter plusieurs pilotes et créer des classes de stockage qui pointent vers l'un ou l'autre. Par exemple, vous pouvez configurer un `san-dev` classe qui utilise le `ontap-`

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

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

Authentification

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

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

Vous pouvez mettre à jour les systèmes back-end existants pour passer d'une méthode basée sur les identifiants à une méthode basée sur les certificats. Toutefois, une seule méthode d'authentification est prise en charge à la fois. Pour passer à une méthode d'authentification différente, vous devez supprimer la méthode existante de la configuration backend.

> Si vous tentez de fournir **les deux identifiants et les certificats**, la création du back-end échoue avec une erreur indiquant que plus d'une méthode d'authentification a été fournie dans le fichier de configuration.

Activer l'authentification basée sur les informations d'identification

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

Voici un exemple de définition du back-end :

YAML

Version: 1 backendName: ExempleBackend storageDriverName: ontap-san managementLIF: 10.0.0.1 svm: svm_nfs username: Vsadmin password: Password

JSON

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

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

Activez l'authentification basée sur les certificats

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

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

Un flux de travail type comprend les étapes suivantes.

Étapes

1. Générez un certificat client et une clé. Lors de la génération, définissez le nom commun (CN) sur l'utilisateur ONTAP pour qu'il s'authentifie.

```
openssl req -x509 -nodes -days 1095 -newkey rsa:2048 -keyout k8senv.key
-out k8senv.pem -subj "/C=US/ST=NC/L=RTP/O=NetApp/CN=admin"
```

2. Ajoutez un certificat d'autorité de certification de confiance au cluster ONTAP. Il se peut déjà que l'administrateur de stockage gère cet espace. Ignorer si aucune autorité de certification approuvée n'est utilisée.

```
security certificate install -type server -cert-name <trusted-ca-cert-
name> -vserver <vserver-name>
ssl modify -vserver <vserver-name> -server-enabled true -client-enabled
true -common-name <common-name> -serial <SN-from-trusted-CA-cert> -ca
<cert-authority>
```

3. Installez le certificat client et la clé (à partir de l'étape 1) sur le cluster ONTAP.

```
security certificate install -type client-ca -cert-name <certificate-
name> -vserver <vserver-name>
security ssl modify -vserver <vserver-name> -client-enabled true
```

4. Vérifiez que le rôle de connexion de sécurité ONTAP est pris en charge `cert` methode d'authentification.

```
security login create -user-or-group-name admin -application ontapi
-authentication-method cert
security login create -user-or-group-name admin -application http
-authentication-method cert
```

5. Testez l'authentification à l'aide d'un certificat généré. Remplacer <ONTAP Management LIF> et <vserver name> par Management LIF IP et SVM name.

```
curl -X POST -Lk https://<ONTAP-Management-
LIF>/servlets/netapp.servlets.admin.XMLrequest_filer --key k8senv.key
--cert ~/k8senv.pem -d '<?xml version="1.0" encoding="UTF-8"?><netapp
xmlns="http://www.netapp.com/filer/admin" version="1.21"
vfiler="<vserver-name>"><vserver-get></vserver-get></netapp>'
```

6. Encodez le certificat, la clé et le certificat CA de confiance avec Base64.

```
base64 -w 0 k8senv.pem >> cert_base64
base64 -w 0 k8senv.key >> key_base64
base64 -w 0 trustedca.pem >> trustedca_base64
```

7. Créez le back-end à l'aide des valeurs obtenues à partir de l'étape précédente.

```
cat cert-backend.json
{
"version": 1,
"storageDriverName": "ontap-san",
"backendName": "SanBackend",
"managementLIF": "1.2.3.4",
"svm": "vserver_test",
"clientCertificate": "Faaaakkkkeeee...Vaaalllluuuueeee",
"clientPrivateKey": "LS0tFaKE...0VaLuES0tLS0K",
"trustedCACertificate": "QNFinfO...SiqOyN",
"storagePrefix": "myPrefix_"
}

tridentctl create backend -f cert-backend.json -n trident
+------------+----------------+--------------------------------------
+--------+---------+
|    NAME    | STORAGE DRIVER |                 UUID                 |
STATE  | VOLUMES |
+------------+----------------+--------------------------------------
+--------+---------+
| SanBackend | ontap-san      | 586b1cd5-8cf8-428d-a76c-2872713612c1 |
online |       0 |
+------------+----------------+--------------------------------------
+--------+---------+
```

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

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

```
cat cert-backend-updated.json
{
"version": 1,
"storageDriverName": "ontap-san",
"backendName": "SanBackend",
"managementLIF": "1.2.3.4",
"svm": "vserver_test",
"username": "vsadmin",
"password": "password",
"storagePrefix": "myPrefix_"
}

#Update backend with tridentctl
tridentctl update backend SanBackend -f cert-backend-updated.json -n
trident
+------------+----------------+--------------------------------------
+--------+---------+
|    NAME    | STORAGE DRIVER |                 UUID                 |
STATE  | VOLUMES |
+------------+----------------+--------------------------------------
+--------+---------+
| SanBackend | ontap-san      | 586b1cd5-8cf8-428d-a76c-2872713612c1 |
online |       9 |
+------------+----------------+--------------------------------------
+--------+---------+
```

Lors de la rotation des mots de passe, l'administrateur du stockage doit d'abord mettre à jour le mot de passe de l'utilisateur sur ONTAP. Cette opération est suivie d'une mise à jour du back-end. Lors de la rotation de certificats, plusieurs certificats peuvent être ajoutés à l'utilisateur. Le back-end est ensuite mis à jour pour utiliser le nouveau certificat, en suivant lequel l'ancien certificat peut être supprimé du cluster ONTAP.

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

Spécifiez les igroups

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

- Astra Trident peut créer et gérer automatiquement un groupe initiateur par système back-end. Si `igroupName` N'est pas inclus dans la définition du système back-end, Astra Trident crée un groupe initiateur nommé `trident-<backend-UUID>` Sur le SVM. Cela permet de s'assurer que chaque système back-end dispose d'un groupe initiateur dédié et de gérer l'ajout/la suppression automatiques d'IQN de nœud Kubernetes.
- Alternativement, les igroups pré-créés peuvent être fournis dans une définition de back-end. Pour ce faire,

utilisez le `igroupName` paramètre config. Astra Trident ajoute/supprime des IQN de nœud Kubernetes au groupe initiateur préexistant.

Pour les systèmes back-end dont ils ont besoin `igroupName` défini, le `igroupName` peut être supprimé avec un `tridentctl backend update` Pour bénéficier des igroups à manipulation automatique avec Astra Trident. L'accès aux volumes déjà rattachés aux charges de travail ne sera pas perturbé. Les futures connexions seront gérées à l'aide du groupe initiateur Astra Trident.

> Dédier un groupe initiateur à chaque instance unique d'Astra Trident est une bonne pratique bénéfique pour l'administrateur Kubernetes et l'administrateur du stockage. CSI Trident automatise l'ajout et la suppression des IQN du nœud du cluster au groupe initiateur, ce qui simplifie considérablement sa gestion. Lorsque vous utilisez le même SVM sur tous les environnements Kubernetes (et avec des installations Trident d'Astra), un groupe initiateur dédié permet de s'assurer que les modifications apportées à un cluster Kubernetes n'influencent pas les groupes initiateurs associés à un autre. En outre, il est important de s'assurer que chaque nœud du cluster Kubernetes dispose d'un IQN unique. Comme mentionné ci-dessus, Astra Trident s'occupe automatiquement de l'ajout et de la suppression des IQN. La réutilisation d'IQN sur des hôtes peut entraîner des scénarios indésirables où les hôtes se confondu les uns avec les autres et où l'accès aux LUN est refusé.

Si Astra Trident est configuré pour fonctionner comme un provisionnement CSI, les IQN du nœud Kubernetes sont automatiquement ajoutés ou supprimés du groupe initiateur. Lorsque des nœuds sont ajoutés à un cluster Kubernetes, `trident-csi` DemonSet déploie un pod (`trident-csi-xxxxx` dans les versions antérieures à 23.01 ou `trident-node<operating system>-xxxx` dans 23.01 et versions ultérieures) sur les nouveaux nœuds ajoutés et enregistre les nouveaux nœuds sur lesquels il peut attacher des volumes. Les IQN du nœud sont également ajoutés au groupe initiateur du back-end. Un ensemble d'étapes similaire gère la suppression des IQN lorsque le(s) nœud(s) est cordeleted, drainé et supprimé de Kubernetes.

Si Astra Trident ne s'exécute pas comme un provisionnement CSI, le groupe initiateur doit être mis à jour manuellement pour contenir les IQN iSCSI de chaque nœud worker du cluster Kubernetes. Les IQN des nœuds qui rejoignent le cluster Kubernetes devront être ajoutés au groupe initiateur. De même, les IQN des nœuds qui sont supprimés du cluster Kubernetes doivent être supprimés du groupe initiateur.

Authentifier les connexions avec le protocole CHAP bidirectionnel

Astra Trident peut authentifier les sessions iSCSI avec le protocole CHAP bidirectionnel pour le `ontap-san` et `ontap-san-economy` pilotes. Pour cela, il faut activer `useCHAP` dans votre définition backend. Lorsqu'il est réglé sur `true`, Astra Trident configure la sécurité de l'initiateur par défaut du SVM en CHAP bidirectionnel et définit le nom d'utilisateur et les secrets du fichier backend. NetApp recommande d'utiliser le protocole CHAP bidirectionnel pour l'authentification des connexions. Voir l'exemple de configuration suivant :

```
---
version: 1
storageDriverName: ontap-san
backendName: ontap_san_chap
managementLIF: 192.168.0.135
svm: ontap_iscsi_svm
useCHAP: true
username: vsadmin
password: password
igroupName: trident
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSd6cNwxyz
```

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

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

Comment cela fonctionne

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

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

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

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

Vous pouvez mettre à jour les informations d'identification CHAP en mettant à jour les paramètres CHAP dans le `backend.json` fichier. Cela nécessitera la mise à jour des secrets CHAP et l'utilisation de `tridentctl`

`update` pour refléter ces modifications.

> Lors de la mise à jour des secrets CHAP pour un back-end, vous devez utiliser `tridentctl` pour mettre à jour le backend. Ne mettez pas à jour les identifiants du cluster de stockage via l'interface de ligne de commande/ONTAP car Astra Trident ne pourra pas détecter ces modifications.

```
cat backend-san.json
{
    "version": 1,
    "storageDriverName": "ontap-san",
    "backendName": "ontap_san_chap",
    "managementLIF": "192.168.0.135",
    "svm": "ontap_iscsi_svm",
    "useCHAP": true,
    "username": "vsadmin",
    "password": "password",
    "igroupName": "trident",
    "chapInitiatorSecret": "cl9qxUpDaTeD",
    "chapTargetInitiatorSecret": "rqxigXgkeUpDaTeD",
    "chapTargetUsername": "iJF4heBRT0TCwxyz",
    "chapUsername": "uh2aNCLSd6cNwxyz",
}

./tridentctl update backend ontap_san_chap -f backend-san.json -n trident
+----------------+----------------+--------------------------------------
+--------+---------+
|   NAME         | STORAGE DRIVER |                 UUID                 |
STATE   | VOLUMES |
+----------------+----------------+--------------------------------------
+--------+---------+
| ontap_san_chap | ontap-san      | aa458f3b-ad2d-4378-8a33-1a472ffbeb5c |
online |       7 |
+----------------+----------------+--------------------------------------
+--------+---------+
```

Les connexions existantes ne seront pas affectées. Elles restent actives si les identifiants sont mis à jour par Astra Trident sur le SVM. Les nouvelles connexions utiliseront les informations d'identification mises à jour et les connexions existantes continuent de rester actives. La déconnexion et la reconnexion des anciens volumes persistants se traduront par l'utilisation des identifiants mis à jour.

Options et exemples de configuration des SAN ONTAP

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

Options de configuration du back-end

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

Paramètre	Description	Valeur par défaut
`version`		Toujours 1
`storageDriverName`	Nom du pilote de stockage	ontap-nas, ontap-nas-économie, ontap-nas-flexgroup, ontap-san », « ontap-san », « ontap-économie san »
`backendName`	Nom personnalisé ou système back-end de stockage	Nom du pilote + "_" + dataLIF
`managementLIF`	Adresse IP d'un cluster ou d'une LIF de gestion SVM pour un basculement MetroCluster transparent, vous devez spécifier une LIF de gestion SVM. Un nom de domaine complet (FQDN) peut être spécifié. Peut être configuré pour utiliser des adresses IPv6 si Astra Trident a été installé à l'aide du `--use-ipv6` drapeau. Les adresses IPv6 doivent être définies entre crochets, telles que [28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555].	« 10.0.0.1 », « [2001:1234:abcd::fefe] »
`dataLIF`	Adresse IP de la LIF de protocole. **Ne pas spécifier pour iSCSI.** utilisations d'Astra Trident "Mappage de LUN sélectif ONTAP" Pour découvrir les LIFs iSCI nécessaires à l'établissement d'une session multi-chemin. Un avertissement est généré si `dataLIF` est explicitement défini.	Dérivé par la SVM
`useCHAP`	Utilisez CHAP pour authentifier iSCSI pour les pilotes SAN ONTAP [Boolean]. Réglez sur `true` Pour qu'Astra Trident configure et utilise le protocole CHAP bidirectionnel comme authentification par défaut pour la SVM donnée en back-end. Reportez-vous à la section "Préparez la configuration du système back-end avec les pilotes SAN ONTAP" pour plus d'informations.	faux
`chapInitiatorSecret`	Secret de l'initiateur CHAP. Requis si `useCHAP=true`	« »

Paramètre	Description	Valeur par défaut
`labels`	Ensemble d'étiquettes arbitraires au format JSON à appliquer aux volumes	« »
`chapTargetInitiatorSecret`	Secret de l'initiateur cible CHAP. Requis si `useCHAP=true`	« »
`chapUsername`	Nom d'utilisateur entrant. Requis si `useCHAP=true`	« »
`chapTargetUsername`	Nom d'utilisateur cible. Requis si `useCHAP=true`	« »
`clientCertificate`	Valeur encodée en Base64 du certificat client. Utilisé pour l'authentification par certificat	« »
`clientPrivateKey`	Valeur encodée en Base64 de la clé privée du client. Utilisé pour l'authentification par certificat	« »
`trustedCACertificate`	Valeur encodée en Base64 du certificat CA de confiance. Facultatif. Utilisé pour l'authentification basée sur des certificats.	« »
`username`	Le nom d'utilisateur devait communiquer avec le cluster ONTAP. Utilisé pour l'authentification basée sur les identifiants.	« »
`password`	Mot de passe requis pour communiquer avec le cluster ONTAP. Utilisé pour l'authentification basée sur les identifiants.	« »
`svm`	Serveur virtuel de stockage à utiliser	Dérivé d'un SVM `managementLIF` est spécifié
`igroupName`	Nom du groupe initiateur à utiliser pour les volumes SAN. Reportez-vous à la section pour en savoir plus.	Trident-<backend-UUID>
`storagePrefix`	Préfixe utilisé pour le provisionnement des nouveaux volumes dans la SVM. Ne peut pas être modifié ultérieurement. Pour mettre à jour ce paramètre, vous devez créer un nouveau backend.	trident

Paramètre	Description	Valeur par défaut
`limitAggregateUsage`	Echec du provisionnement si l'utilisation est supérieure à ce pourcentage. Si vous utilisez un système Amazon FSX pour le système back-end NetApp ONTAP, ne spécifiez pas `limitAggregateUsage`. Le fourni `fsxadmin` et `vsadmin` Ne contiennent pas les autorisations requises pour récupérer l'utilisation d'agrégats et le limiter à l'aide d'Astra Trident.	« » (non appliqué par défaut)
`limitVolumeSize`	Echec du provisionnement si la taille du volume demandé est supérieure à cette valeur. Restreint également la taille maximale des volumes qu'il gère pour les qtrees et les LUN.	« » (non appliqué par défaut)
`lunsPerFlexvol`	Nombre maximal de LUN par FlexVol, doit être compris dans la plage [50, 200]	"100"
`debugTraceFlags`	Indicateurs de débogage à utiliser lors du dépannage. Par exemple, {"api":false, "méthode":true} ne pas utiliser sauf si vous effectuez un dépannage et que vous avez besoin d'un vidage détaillé des journaux.	nul

Paramètre	Description	Valeur par défaut
`useREST`	Paramètre booléen pour utiliser les API REST de ONTAP. **Aperçu technique** `useREST` est fourni sous forme d'aperçu technique ** qui est recommandé pour les environnements de test et non pour les charges de travail de production. Lorsqu'il est réglé sur `true`, Astra Trident va utiliser les API REST de ONTAP pour communiquer avec le système back-end. Cette fonctionnalité requiert ONTAP 9.11.1 et versions ultérieures. En outre, le rôle de connexion ONTAP utilisé doit avoir accès au `ontap` client supplémentaire. Ceci est satisfait par le pré-défini `vsadmin` et `cluster-admin` rôles. `useREST` N'est pas pris en charge par MetroCluster.	faux

Détails sur `igroupName`

`igroupName` Peut être défini sur un groupe initiateur déjà créé sur le cluster ONTAP. Si non spécifié, Astra Trident crée automatiquement un groupe initiateur nommé `trident-<backend-UUID>`.

Si vous disposez d'un nom de groupe prédéfini, nous vous recommandons d'utiliser un groupe initiateur par cluster Kubernetes si le SVM doit être partagé entre les environnements. Cela est nécessaire pour qu'Astra Trident conserve automatiquement les ajouts et suppressions d'IQN.

- `igroupName` Peut être mis à jour afin de désigner un nouveau groupe initiateur créé et géré sur la SVM en dehors d'Astra Trident.
- `igroupName` peut être omis. Dans ce cas, Astra Trident crée et gère un groupe initiateur nommé `trident-<backend-UUID>` automatiquement.

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

Options de configuration back-end pour les volumes de provisionnement

Vous pouvez contrôler le provisionnement par défaut à l'aide de ces options dans `defaults` section de la configuration. Pour un exemple, voir les exemples de configuration ci-dessous.

Paramètre	Description	Valeur par défaut
`spaceAllocation`	Allocation d'espace pour les LUN	« vrai »

Paramètre	Description	Valeur par défaut
`spaceReserve`	Mode de réservation d'espace ; "none" (fin) ou "volume" (épais)	« aucun »
`snapshotPolicy`	Règle Snapshot à utiliser	« aucun »
`qosPolicy`	QoS policy group à affecter pour les volumes créés. Choisissez une de qosPolicy ou adaptiveQosPolicy par pool de stockage/back-end. Avec Astra Trident, les groupes de règles de QoS doivent être utilisés avec ONTAP 9.8 ou version ultérieure. Nous recommandons l'utilisation d'un groupe de règles de qualité de service non partagé et nous assurer que le groupe de règles est appliqué à chaque composant individuellement. Un groupe de règles de QoS partagé appliquera le plafond du débit total de toutes les charges de travail.	« »
`adaptiveQosPolicy`	Groupe de règles de QoS adaptative à attribuer aux volumes créés. Choisissez une de qosPolicy ou adaptiveQosPolicy par pool de stockage/back-end	« »
`snapshotReserve`	Pourcentage du volume réservé pour les instantanés "0"	Si `snapshotPolicy` est « aucun », sinon « »
`splitOnClone`	Séparer un clone de son parent lors de sa création	« faux »
`encryption`	Activez NetApp Volume Encryption (NVE) sur le nouveau volume. La valeur par défaut est `false`. Pour utiliser cette option, NVE doit être sous licence et activé sur le cluster. Si NAE est activé sur le back-end, tous les volumes provisionnés dans Astra Trident seront activés par NAE. Pour plus d'informations, se reporter à : "Fonctionnement d'Astra Trident avec NVE et NAE".	« faux »
`luksEncryption`	Activez le cryptage LUKS. Reportez-vous à la section "Utiliser la configuration de clé unifiée Linux (LUKS)".	« »
`securityStyle`	Style de sécurité pour les nouveaux volumes	`unix`

Paramètre	Description	Valeur par défaut
`tieringPolicy`	La stratégie de hiérarchisation à utiliser « none »	Snapshot uniquement pour une configuration SVM-DR pré-ONTAP 9.5

Exemples de provisionnement de volumes

Voici un exemple avec des valeurs par défaut définies :

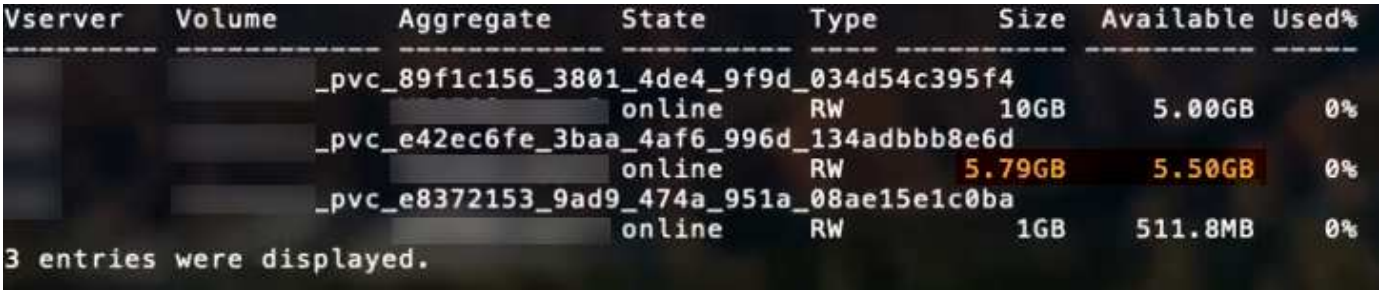
```
---
version: 1
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: trident_svm
username: admin
password: password
labels:
  k8scluster: dev2
  backend: dev2-sanbackend
storagePrefix: alternate-trident
igroupName: custom
debugTraceFlags:
  api: false
  method: true
defaults:
  spaceReserve: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

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

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

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

Le modèle 1.1 est le modèle 10 % d'Astra Trident supplémentaire qui s'ajoute à la baie FlexVol pour prendre en charge les métadonnées de la LUN. Pour `snapshotReserve` = 5 % et demande de volume persistant = 5 Gio, la taille totale du volume est de 5,7 Gio et la taille disponible est de 5,5 Gio. Le `volume show` la commande doit afficher des résultats similaires à cet exemple :



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

Exemples de configuration minimaux

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

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

`ontap-san` pilote avec authentification par certificat

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

```
---
version: 1
storageDriverName: ontap-san
backendName: DefaultSANBackend
managementLIF: 10.0.0.1
svm: svm_iscsi
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSd6cNwxyz
igroupName: trident
clientCertificate: ZXR0ZXJwYXB...ICMgJ3BhcGVyc2
clientPrivateKey: vciwKIyAgZG...0cnksIGRlc2NyaX
trustedCACertificate: zcyBbaG...b3Igb3duIGNsYXNz
```

`ontap-san` Pilote avec CHAP bidirectionnel

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

```
---
version: 1
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_iscsi
labels:
  k8scluster: test-cluster-1
  backend: testcluster1-sanbackend
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSd6cNwxyz
igroupName: trident
username: vsadmin
password: password
```

`ontap-san-economy` conducteur

```
---
version: 1
storageDriverName: ontap-san-economy
managementLIF: 10.0.0.1
svm: svm_iscsi_eco
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSd6cNwxyz
igroupName: trident
username: vsadmin
password: password
```

Exemples de systèmes back-end avec pools virtuels

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

Astra Trident définit les étiquettes de provisionnement dans le champ « Commentaires ». Les commentaires sont définis sur le FlexVol. Astra Trident copie toutes les étiquettes présentes sur un pool virtuel vers le volume

de stockage lors du provisionnement. Pour plus de commodité, les administrateurs du stockage peuvent définir des étiquettes par pool virtuel et les volumes de groupe par étiquette.

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

```
---
version: 1
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_iscsi
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSd6cNwxyz
igroupName: trident
username: vsadmin
password: password
defaults:
  spaceAllocation: 'false'
  encryption: 'false'
  qosPolicy: standard
labels:
  store: san_store
  kubernetes-cluster: prod-cluster-1
region: us_east_1
storage:
- labels:
    protection: gold
    creditpoints: '40000'
  zone: us_east_1a
  defaults:
    spaceAllocation: 'true'
    encryption: 'true'
    adaptiveQosPolicy: adaptive-extreme
- labels:
    protection: silver
    creditpoints: '20000'
  zone: us_east_1b
  defaults:
    spaceAllocation: 'false'
    encryption: 'true'
    qosPolicy: premium
- labels:
    protection: bronze
    creditpoints: '5000'
  zone: us_east_1c
  defaults:
    spaceAllocation: 'true'
    encryption: 'false'
```

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

```
---
version: 1
storageDriverName: ontap-san-economy
managementLIF: 10.0.0.1
svm: svm_iscsi_eco
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSd6cNwxyz
igroupName: trident
username: vsadmin
password: password
defaults:
  spaceAllocation: 'false'
  encryption: 'false'
labels:
  store: san_economy_store
region: us_east_1
storage:
- labels:
    app: oracledb
    cost: '30'
  zone: us_east_1a
  defaults:
    spaceAllocation: 'true'
    encryption: 'true'
- labels:
    app: postgresdb
    cost: '20'
  zone: us_east_1b
  defaults:
    spaceAllocation: 'false'
    encryption: 'true'
- labels:
    app: mysqldb
    cost: '10'
  zone: us_east_1c
  defaults:
    spaceAllocation: 'true'
    encryption: 'false'
```

Mappage des systèmes back-end aux classes de stockage

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

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

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

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

Configurer un système NAS backend ONTAP

Découvrez comment configurer un back-end ONTAP avec les pilotes ONTAP et NAS Cloud Volumes ONTAP.

- "Préparation"
- "Configuration et exemples"

> Astra Control assure une protection, une reprise d'activité et une mobilité transparentes (en déplaçant des volumes entre les clusters Kubernetes) pour les volumes créés avec le système `ontap-nas`, `ontap-nas-flexgroup`, et `ontap-san` pilotes. Voir "Conditions préalables à la réplication d'Astra Control" pour plus d'informations.
>
> - Vous devez utiliser `ontap-nas` adapté aux charges de travail de production qui nécessitent une protection des données, une reprise d'activité et la mobilité.
> - Utiliser `ontap-san-economy` Lorsque vous prévoyez une utilisation de volume, celle-ci devrait être bien supérieure à celle prise en charge par ONTAP.
> - Utiliser `ontap-nas-economy` Ce n'est que lorsque l'utilisation prévue des volumes sera beaucoup plus élevée que ce que prend en charge ONTAP, et le `ontap-san-economy` le pilote ne peut pas être utilisé.
> - Ne pas utiliser `ontap-nas-economy` si vous prévoyez d'avoir besoin en termes de protection des données, de reprise sur incident ou de mobilité.

Autorisations utilisateur

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

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

S'il est possible de créer un rôle plus restrictif au sein de ONTAP qu'un pilote Trident peut utiliser, nous ne le recommandons pas. La plupart des nouvelles versions de Trident appellent des API supplémentaires qui devront être prises en compte, ce qui complique les mises à niveau et risque d'erreurs.

Préparez la configuration d'un système back-end avec les pilotes NAS ONTAP

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

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

N'oubliez pas que vous pouvez également exécuter plusieurs pilotes et créer des classes de stockage qui pointent vers l'un ou l'autre. Par exemple, vous pouvez configurer une classe Gold qui utilise le `ontap-nas` Pilote et une classe Bronze qui utilise le `ontap-nas-economy` une seule.

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

Authentification

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

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

Vous pouvez mettre à jour les systèmes back-end existants pour passer d'une méthode basée sur les identifiants à une méthode basée sur les certificats. Toutefois, une seule méthode d'authentification est prise en charge à la fois. Pour passer à une méthode d'authentification différente, vous devez supprimer la méthode existante de la configuration backend.

Si vous tentez de fournir **les deux identifiants et les certificats**, la création du back-end échoue avec une erreur indiquant que plus d'une méthode d'authentification a été fournie dans le fichier de configuration.

Activer l'authentification basée sur les informations d'identification

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

Voici un exemple de définition du back-end :

YAML

```
---
version: 1
backendName: ExampleBackend
storageDriverName: ontap-nas
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
svm: svm_nfs
username: vsadmin
password: password
```

JSON

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

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

Activez l'authentification basée sur les certificats

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

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

Un flux de travail type comprend les étapes suivantes.

Étapes

1. Générez un certificat client et une clé. Lors de la génération, définissez le nom commun (CN) sur

l'utilisateur ONTAP pour qu'il s'authentifie.

```
openssl req -x509 -nodes -days 1095 -newkey rsa:2048 -keyout k8senv.key
-out k8senv.pem -subj "/C=US/ST=NC/L=RTP/O=NetApp/CN=vsadmin"
```

2. Ajoutez un certificat d'autorité de certification de confiance au cluster ONTAP. Il se peut déjà que l'administrateur de stockage gère cet espace. Ignorer si aucune autorité de certification approuvée n'est utilisée.

```
security certificate install -type server -cert-name <trusted-ca-cert-
name> -vserver <vserver-name>
ssl modify -vserver <vserver-name> -server-enabled true -client-enabled
true -common-name <common-name> -serial <SN-from-trusted-CA-cert> -ca
<cert-authority>
```

3. Installez le certificat client et la clé (à partir de l'étape 1) sur le cluster ONTAP.

```
security certificate install -type client-ca -cert-name <certificate-
name> -vserver <vserver-name>
security ssl modify -vserver <vserver-name> -client-enabled true
```

4. Vérifiez que le rôle de connexion de sécurité ONTAP est pris en charge `cert` methode d'authentification.

```
security login create -user-or-group-name vsadmin -application ontapi
-authentication-method cert -vserver <vserver-name>
security login create -user-or-group-name vsadmin -application http
-authentication-method cert -vserver <vserver-name>
```

5. Testez l'authentification à l'aide d'un certificat généré. Remplacer <ONTAP Management LIF> et <vserver name> par Management LIF IP et SVM name. Vous devez vous assurer que le LIF a sa politique de service définie sur `default-data-management`.

```
curl -X POST -Lk https://<ONTAP-Management-
LIF>/servlets/netapp.servlets.admin.XMLrequest_filer --key k8senv.key
--cert ~/k8senv.pem -d '<?xml version="1.0" encoding="UTF-8"?><netapp
xmlns="http://www.netapp.com/filer/admin" version="1.21"
vfiler="<vserver-name>"><vserver-get></vserver-get></netapp>'
```

6. Encodez le certificat, la clé et le certificat CA de confiance avec Base64.

```
base64 -w 0 k8senv.pem >> cert_base64
base64 -w 0 k8senv.key >> key_base64
base64 -w 0 trustedca.pem >> trustedca_base64
```

7. Créez le back-end à l'aide des valeurs obtenues à partir de l'étape précédente.

```
cat cert-backend-updated.json
{
"version": 1,
"storageDriverName": "ontap-nas",
"backendName": "NasBackend",
"managementLIF": "1.2.3.4",
"dataLIF": "1.2.3.8",
"svm": "vserver_test",
"clientCertificate": "Faaaakkkkeeee...Vaaalllluuuueeee",
"clientPrivateKey": "LS0tFaKE...0VaLuES0tLS0K",
"storagePrefix": "myPrefix_"
}

#Update backend with tridentctl
tridentctl update backend NasBackend -f cert-backend-updated.json -n
trident
+------------+----------------+--------------------------------------
+--------+---------+
|    NAME    | STORAGE DRIVER |                 UUID                 |
STATE  | VOLUMES |
+------------+----------------+--------------------------------------
+--------+---------+
| NasBackend | ontap-nas      | 98e19b74-aec7-4a3d-8dcf-128e5033b214 |
online |       9 |
+------------+----------------+--------------------------------------
+--------+---------+
```

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

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

```
cat cert-backend-updated.json
{
"version": 1,
"storageDriverName": "ontap-nas",
"backendName": "NasBackend",
"managementLIF": "1.2.3.4",
"dataLIF": "1.2.3.8",
"svm": "vserver_test",
"username": "vsadmin",
"password": "password",
"storagePrefix": "myPrefix_"
}

#Update backend with tridentctl
tridentctl update backend NasBackend -f cert-backend-updated.json -n
trident
+------------+----------------+--------------------------------------
+--------+---------+
|    NAME    | STORAGE DRIVER |                 UUID                 |
STATE  | VOLUMES |
+------------+----------------+--------------------------------------
+--------+---------+
| NasBackend | ontap-nas      | 98e19b74-aec7-4a3d-8dcf-128e5033b214 |
online |       9 |
+------------+----------------+--------------------------------------
+--------+---------+
```

Lors de la rotation des mots de passe, l'administrateur du stockage doit d'abord mettre à jour le mot de passe de l'utilisateur sur ONTAP. Cette opération est suivie d'une mise à jour du back-end. Lors de la rotation de certificats, plusieurs certificats peuvent être ajoutés à l'utilisateur. Le back-end est ensuite mis à jour pour utiliser le nouveau certificat, en suivant lequel l'ancien certificat peut être supprimé du cluster ONTAP.

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

Gestion des règles d'exportation NFS

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

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

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

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

Gérez les règles d'exportation de manière dynamique

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

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

Exemple

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

```
---
version: 1
storageDriverName: ontap-nas
backendName: ontap_nas_auto_export
managementLIF: 192.168.0.135
svm: svm1
username: vsadmin
password: password
autoExportCIDRs:
- 192.168.0.0/24
autoExportPolicy: true
```

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

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

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

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

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

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

```
./tridentctl get backends ontap_nas_auto_export -n trident -o yaml
items:
- backendUUID: 403b5326-8482-40db-96d0-d83fb3f4daec
  config:
    aggregate: ""
    autoExportCIDRs:
    - 192.168.0.0/24
    autoExportPolicy: true
    backendName: ontap_nas_auto_export
    chapInitiatorSecret: ""
    chapTargetInitiatorSecret: ""
    chapTargetUsername: ""
    chapUsername: ""
    dataLIF: 192.168.0.135
    debug: false
    debugTraceFlags: null
    defaults:
      encryption: "false"
      exportPolicy: <automatic>
      fileSystemType: ext4
```

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

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

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

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

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

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

Options et exemples de configuration du NAS ONTAP

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

Options de configuration du back-end

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

Paramètre	Description	Valeur par défaut
`version`		Toujours 1
`storageDriverName`	Nom du pilote de stockage	ontap-nas, ontap-nas-économie, ontap-nas-flexgroup, ontap-san », « ontap-san », « ontap-économie san »
`backendName`	Nom personnalisé ou système back-end de stockage	Nom du pilote + "_" + dataLIF
`managementLIF`	Adresse IP d'un cluster ou d'une LIF de gestion SVM pour un basculement MetroCluster transparent, vous devez spécifier une LIF de gestion SVM. Un nom de domaine complet (FQDN) peut être spécifié. Peut être configuré pour utiliser des adresses IPv6 si Astra Trident a été installé à l'aide du `--use-ipv6` drapeau. Les adresses IPv6 doivent être définies entre crochets, telles que [28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555].	« 10.0.0.1 », « [2001:1234:abcd::fefe] »

Paramètre	Description	Valeur par défaut
`dataLIF`	Adresse IP de la LIF de protocole. Nous vous recommandons de spécifier `dataLIF`. Si elle n'est pas fournie, Astra Trident extrait les LIF de données du SVM. Vous pouvez spécifier un nom de domaine complet (FQDN) à utiliser pour les opérations de montage NFS, permettant de créer un DNS Round-Robin pour équilibrer la charge sur plusieurs LIF de données. Peut être modifié après le réglage initial. Reportez-vous à la section . Peut être configuré pour utiliser des adresses IPv6 si Astra Trident a été installé à l'aide du `--use-ipv6` drapeau. Les adresses IPv6 doivent être définies entre crochets, telles que [28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555].	Adresse spécifiée ou dérivée d'un SVM, si non spécifiée (non recommandé)
`autoExportPolicy`	Activer la création et la mise à jour automatiques des règles d'exportation [booléennes]. À l'aide du `autoExportPolicy` et `autoExportCIDRs` Avec Astra Trident, il peut gérer automatiquement les règles d'exportation.	faux
`autoExportCIDRs`	Liste des CIDR pour filtrer les adresses IP du nœud Kubernetes par rapport à quand `autoExportPolicy` est activé. À l'aide du `autoExportPolicy` et `autoExportCIDRs` Avec Astra Trident, il peut gérer automatiquement les règles d'exportation.	["0.0.0.0/0", "::/0"]"
`labels`	Ensemble d'étiquettes arbitraires au format JSON à appliquer aux volumes	« »
`clientCertificate`	Valeur encodée en Base64 du certificat client. Utilisé pour l'authentification par certificat	« »
`clientPrivateKey`	Valeur encodée en Base64 de la clé privée du client. Utilisé pour l'authentification par certificat	« »

Paramètre	Description	Valeur par défaut
`trustedCACertificate`	Valeur encodée en Base64 du certificat CA de confiance. Facultatif. Utilisé pour l'authentification par certificat	« »
`username`	Nom d'utilisateur pour la connexion au cluster/SVM. Utilisé pour l'authentification basée sur les identifiants	
`password`	Mot de passe pour la connexion au cluster/SVM. Utilisé pour l'authentification basée sur les identifiants	
`svm`	Serveur virtuel de stockage à utiliser	Dérivé d'un SVM `managementLIF` est spécifié
`storagePrefix`	Préfixe utilisé pour le provisionnement des nouveaux volumes dans la SVM. Ne peut pas être mis à jour une fois que vous l'avez défini	trident
`limitAggregateUsage`	Echec du provisionnement si l'utilisation est supérieure à ce pourcentage. **Ne s'applique pas à Amazon FSX pour ONTAP**	« » (non appliqué par défaut)
`limitVolumeSize`	Echec du provisionnement si la taille du volume demandé est supérieure à cette valeur.	« » (non appliqué par défaut)
`limitVolumeSize`	Echec du provisionnement si la taille du volume demandé est supérieure à cette valeur. Restreint également la taille maximale des volumes qu'il gère pour les qtrees et les LUN, et la `qtreesPerFlexvol` L'option permet de personnaliser le nombre maximal de qtree par FlexVol.	« » (non appliqué par défaut)
`lunsPerFlexvol`	Nombre maximal de LUN par FlexVol, doit être compris dans la plage [50, 200]	"100"
`debugTraceFlags`	Indicateurs de débogage à utiliser lors du dépannage. Par exemple, {"api":false, "méthode":true} ne pas utiliser `debugTraceFlags` à moins que vous ne soyez en mesure de dépanner et que vous ayez besoin d'un vidage détaillé des journaux.	nul

Paramètre	Description	Valeur par défaut
`nfsMountOptions`	Liste des options de montage NFS séparée par des virgules. Les options de montage des volumes Kubernetes persistants sont généralement spécifiées dans les classes de stockage, mais si aucune option de montage n'est spécifiée dans une classe de stockage, Astra Trident utilisera les options de montage spécifiées dans le fichier de configuration du système back-end. Si aucune option de montage n'est spécifiée dans la classe de stockage ou le fichier de configuration, Astra Trident ne définit aucune option de montage sur un volume persistant associé.	« »
`qtreesPerFlexvol`	Nombre maximal de qtrees par FlexVol, qui doit être compris dans la plage [50, 300]	"200"
`useREST`	Paramètre booléen pour utiliser les API REST de ONTAP. **Aperçu technique** `useREST` est fourni sous forme d'aperçu technique ** qui est recommandé pour les environnements de test et non pour les charges de travail de production. Lorsqu'il est réglé sur `true`, Astra Trident va utiliser les API REST de ONTAP pour communiquer avec le système back-end. Cette fonctionnalité requiert ONTAP 9.11.1 et versions ultérieures. En outre, le rôle de connexion ONTAP utilisé doit avoir accès au `ontap` client supplémentaire. Ceci est satisfait par le pré-défini `vsadmin` et `cluster-admin` rôles. `useREST` N'est pas pris en charge par MetroCluster.	faux

Options de configuration back-end pour les volumes de provisionnement

Vous pouvez contrôler le provisionnement par défaut à l'aide de ces options dans `defaults` section de la configuration. Pour un exemple, voir les exemples de configuration ci-dessous.

Paramètre	Description	Valeur par défaut
`spaceAllocation`	Allocation d'espace pour les LUN	« vrai »
`spaceReserve`	Mode de réservation d'espace ; "none" (fin) ou "volume" (épais)	« aucun »
`snapshotPolicy`	Règle Snapshot à utiliser	« aucun »
`qosPolicy`	QoS policy group à affecter pour les volumes créés. Choisissez une de qosPolicy ou adaptiveQosPolicy par pool de stockage/back-end	« »
`adaptiveQosPolicy`	Groupe de règles de QoS adaptative à attribuer aux volumes créés. Choisissez une de qosPolicy ou adaptiveQosPolicy par pool de stockage/back-end. Non pris en charge par l'économie ontap-nas.	« »
`snapshotReserve`	Pourcentage du volume réservé pour les instantanés "0"	Si `snapshotPolicy` est « aucun », sinon « »
`splitOnClone`	Séparer un clone de son parent lors de sa création	« faux »
`encryption`	Activez NetApp Volume Encryption (NVE) sur le nouveau volume. La valeur par défaut est `false`. Pour utiliser cette option, NVE doit être sous licence et activé sur le cluster. Si NAE est activé sur le back-end, tous les volumes provisionnés dans Astra Trident seront activés par NAE. Pour plus d'informations, se reporter à : "Fonctionnement d'Astra Trident avec NVE et NAE".	« faux »
`tieringPolicy`	La stratégie de hiérarchisation à utiliser « none »	Snapshot uniquement pour une configuration SVM-DR pré-ONTAP 9.5
`unixPermissions`	Mode pour les nouveaux volumes	"777" pour volumes NFS ; vide (non applicable) pour volumes SMB
`snapshotDir`	Contrôle la visibilité du `.snapshot` répertoire	« faux »
`exportPolicy`	Export policy à utiliser	« par défaut »
`securityStyle`	Style de sécurité pour les nouveaux volumes. Prise en charge de NFS `mixed` et `unix` styles de sécurité. SMB prend en charge `mixed` et `ntfs` styles de sécurité.	NFS par défaut est `unix`. SMB par défaut est `ntfs`.

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

Exemples de provisionnement de volumes

Voici un exemple avec des valeurs par défaut définies :

```
---
version: 1
storageDriverName: ontap-nas
backendName: customBackendName
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
labels:
  k8scluster: dev1
  backend: dev1-nasbackend
svm: trident_svm
username: cluster-admin
password: password
limitAggregateUsage: 80%
limitVolumeSize: 50Gi
nfsMountOptions: nfsvers=4
debugTraceFlags:
  api: false
  method: true
defaults:
  spaceReserve: volume
  qosPolicy: premium
  exportPolicy: myk8scluster
  snapshotPolicy: default
  snapshotReserve: '10'
```

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

Le calcul est le suivant :

```
Total volume size = (PVC requested size) / (1 - (snapshotReserve
percentage) / 100)
```

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

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

Exemples

Exemples de configuration minimaux

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

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

Options par défaut activées <code>ontap-nas-economy</code>

```
---
version: 1
storageDriverName: ontap-nas-economy
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
svm: svm_nfs
username: vsadmin
password: password
```

Authentification basée sur des certificats

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

```
---
version: 1
backendName: DefaultNASBackend
storageDriverName: ontap-nas
managementLIF: 10.0.0.1
dataLIF: 10.0.0.15
svm: nfs_svm
clientCertificate: ZXR0ZXJwYXB...ICMgJ3BhcGVyc2
clientPrivateKey: vciwKIyAgZG...0cnksIGRlc2NyaX
trustedCACertificate: zcyBbaG...b3Igb3duIGNsYXNz
storagePrefix: myPrefix_
```

Export policy auto

Ces exemples vous montrent comment vous pouvez demander à Astra Trident d'utiliser des règles d'exportation dynamiques pour créer et gérer automatiquement les règles d'exportation. Cela fonctionne de la même manière pour le `ontap-nas-economy` et `ontap-nas-flexgroup` pilotes.

pilote ontap-nas

```
---
version: 1
storageDriverName: ontap-nas
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
svm: svm_nfs
labels:
  k8scluster: test-cluster-east-1a
  backend: test1-nasbackend
autoExportPolicy: true
autoExportCIDRs:
- 10.0.0.0/24
username: admin
password: password
nfsMountOptions: nfsvers=4
```

<code>ontap-nas-flexgroup</code> conducteur

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

Utilisation des adresses IPv6

Cet exemple montre `managementLIF` Utilisation d'une adresse IPv6.

```
---
version: 1
storageDriverName: ontap-nas
backendName: nas_ipv6_backend
managementLIF: "[5c5d:5edf:8f:7657:bef8:109b:1b41:d491]"
labels:
  k8scluster: test-cluster-east-1a
  backend: test1-ontap-ipv6
svm: nas_ipv6_svm
username: vsadmin
password: password
```

`ontap-nas-economy` conducteur

```
---
version: 1
storageDriverName: ontap-nas-economy
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
svm: svm_nfs
username: vsadmin
password: password
```

`ontap-nas` Pilote pour Amazon FSX pour ONTAP utilisant des volumes SMB

```
---
version: 1
backendName: SMBBackend
storageDriverName: ontap-nas
managementLIF: example.mgmt.fqdn.aws.com
nasType: smb
dataLIF: 10.0.0.15
svm: nfs_svm
clientCertificate: ZXR0ZXJwYXB...ICMgJ3BhcGVyc2
clientPrivateKey: vciwKIyAgZG...0cnksIGRlc2NyaX
trustedCACertificate: zcyBbaG...b3Igb3duIGNsYXNz
storagePrefix: myPrefix_
```

Exemples de systèmes back-end avec pools virtuels

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

Astra Trident définit les étiquettes de provisionnement dans le champ « Commentaires ». Les commentaires sont définis sur FlexVol pour `ontap-nas` Ou FlexGroup pour `ontap-nas-flexgroup`. Astra Trident copie toutes les étiquettes présentes sur un pool virtuel vers le volume de stockage lors du provisionnement. Pour plus de commodité, les administrateurs du stockage peuvent définir des étiquettes par pool virtuel et les volumes de groupe par étiquette.

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

<code>ontap-nas</code> conducteur

```
---
version: 1
storageDriverName: ontap-nas
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
svm: svm_nfs
username: admin
password: password
nfsMountOptions: nfsvers=4
defaults:
  spaceReserve: none
  encryption: 'false'
  qosPolicy: standard
labels:
  store: nas_store
  k8scluster: prod-cluster-1
region: us_east_1
storage:
- labels:
    app: msoffice
    cost: '100'
  zone: us_east_1a
  defaults:
    spaceReserve: volume
    encryption: 'true'
    unixPermissions: '0755'
    adaptiveQosPolicy: adaptive-premium
- labels:
    app: slack
    cost: '75'
  zone: us_east_1b
  defaults:
    spaceReserve: none
    encryption: 'true'
    unixPermissions: '0755'
- labels:
    app: wordpress
    cost: '50'
  zone: us_east_1c
  defaults:
    spaceReserve: none
    encryption: 'true'
    unixPermissions: '0775'
- labels:
```

```
    app: mysqldb
    cost: '25'
  zone: us_east_1d
  defaults:
    spaceReserve: volume
    encryption: 'false'
    unixPermissions: '0775'
```

<code>ontap-nas-flexgroup</code> conducteur

```
---
version: 1
storageDriverName: ontap-nas-flexgroup
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
svm: svm_nfs
username: vsadmin
password: password
defaults:
  spaceReserve: none
  encryption: 'false'
labels:
  store: flexgroup_store
  k8scluster: prod-cluster-1
region: us_east_1
storage:
- labels:
    protection: gold
    creditpoints: '50000'
  zone: us_east_1a
  defaults:
    spaceReserve: volume
    encryption: 'true'
    unixPermissions: '0755'
- labels:
    protection: gold
    creditpoints: '30000'
  zone: us_east_1b
  defaults:
    spaceReserve: none
    encryption: 'true'
    unixPermissions: '0755'
- labels:
    protection: silver
    creditpoints: '20000'
  zone: us_east_1c
  defaults:
    spaceReserve: none
    encryption: 'true'
    unixPermissions: '0775'
- labels:
    protection: bronze
    creditpoints: '10000'
  zone: us_east_1d
```

```
  defaults:
    spaceReserve: volume
    encryption: 'false'
    unixPermissions: '0775'
```

<code>ontap-nas-economy</code> conducteur

```
---
version: 1
storageDriverName: ontap-nas-economy
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
svm: svm_nfs
username: vsadmin
password: password
defaults:
  spaceReserve: none
  encryption: 'false'
labels:
  store: nas_economy_store
region: us_east_1
storage:
- labels:
    department: finance
    creditpoints: '6000'
  zone: us_east_1a
  defaults:
    spaceReserve: volume
    encryption: 'true'
    unixPermissions: '0755'
- labels:
    department: legal
    creditpoints: '5000'
  zone: us_east_1b
  defaults:
    spaceReserve: none
    encryption: 'true'
    unixPermissions: '0755'
- labels:
    department: engineering
    creditpoints: '3000'
  zone: us_east_1c
  defaults:
    spaceReserve: none
    encryption: 'true'
    unixPermissions: '0775'
- labels:
    department: humanresource
    creditpoints: '2000'
  zone: us_east_1d
  defaults:
```

```
    spaceReserve: volume
    encryption: 'false'
    unixPermissions: '0775'
```

Mise à jour `dataLIF` après la configuration initiale

Vous pouvez modifier la LIF de données après la configuration initiale en exécutant la commande suivante pour fournir le nouveau fichier JSON back-end avec la LIF de données mise à jour.

```
tridentctl update backend <backend-name> -f <path-to-backend-json-file-with-updated-dataLIF>
```

> Si des demandes de volume persistant sont associées à un ou plusieurs pods, tous les pods correspondants doivent être arrêtés, puis réintégrés dans le but de permettre la nouvelle LIF de données d'être effective.

Mappage des systèmes back-end aux classes de stockage

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

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

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

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

Amazon FSX pour NetApp ONTAP

Utilisez Astra Trident avec Amazon FSX pour NetApp ONTAP

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

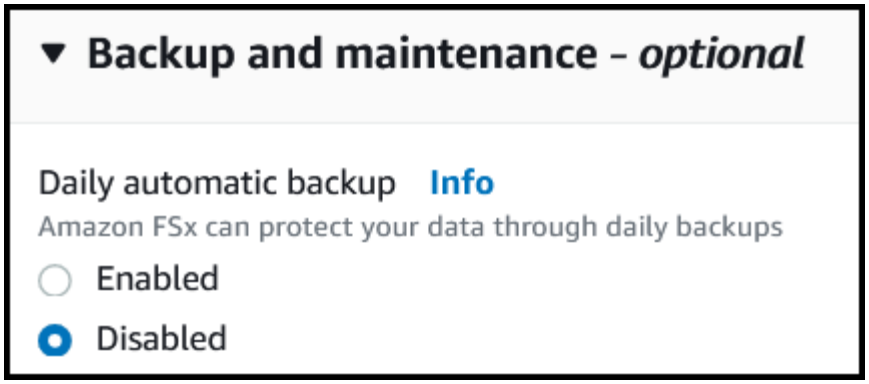
Un système de fichiers est la ressource principale d'Amazon FSX, similaire à un cluster ONTAP sur site. Au sein de chaque SVM, vous pouvez créer un ou plusieurs volumes, qui sont des conteneurs de données qui stockent les fichiers et les dossiers dans votre système de fichiers. Avec Amazon FSX pour NetApp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

Avec Astra Trident avec Amazon FSX pour NetApp ONTAP, vous pouvez vous assurer que les clusters Kubernetes exécutés dans Amazon Elastic Kubernetes Service (EKS) peuvent provisionner des volumes persistants de bloc et de fichier sauvegardés par ONTAP.

Amazon FSX pour NetApp ONTAP utilise "FabricPool" pour gérer les tiers de stockage. Elle vous permet de stocker les données au niveau le plus important, selon que celles-ci sont fréquemment utilisées.

Considérations

- Volumes SMB :
  - Les volumes SMB sont pris en charge à l'aide de `ontap-nas` conducteur uniquement.
  - Astra Trident prend en charge les volumes SMB montés sur des pods qui s'exécutent uniquement sur des nœuds Windows.
  - Astra Trident ne prend pas en charge l'architecture Windows ARM.
- Les volumes créés sur des systèmes de fichiers Amazon FSX dont les sauvegardes automatiques sont activées ne peuvent pas être supprimés par Trident. Pour supprimer des demandes de volume persistant, vous devez supprimer manuellement le volume PV et le volume FSX pour ONTAP. Pour éviter ce problème :
  - N'utilisez pas **création rapide** pour créer le système de fichiers FSX pour ONTAP. Le flux de création rapide active les sauvegardes automatiques et ne propose pas d'option de désinscription.
  - Lorsque vous utilisez **création standard**, désactivez la sauvegarde automatique. La désactivation des sauvegardes automatiques permet à Trident de supprimer un volume sans intervention manuelle supplémentaire.



Pilotes

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

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

Pour plus d'informations sur le pilote, reportez-vous à la section "Pilotes ONTAP".

Authentification

Astra Trident propose deux modes d'authentification.

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

> Astra Trident devrait être exécuté en tant que A. `vsadmin` Utilisateur SVM ou en tant qu'utilisateur avec un nom différent qui a le même rôle. Amazon FSX pour NetApp ONTAP en a un `fsxadmin` Utilisateur qui remplace le ONTAP de manière limitée `admin` utilisateur du cluster. Nous vous recommandons vivement d'utiliser `vsadmin` Avec Astra Trident.

Vous pouvez mettre à jour les systèmes back-end pour passer d'une méthode basée sur les identifiants à une méthode basée sur les certificats. Toutefois, si vous tentez de fournir des identifiants et des certificats *, la création du back-end échouera. Pour passer à une méthode d'authentification différente, vous devez supprimer la méthode existante de la configuration backend.

Pour plus d'informations sur l'activation de l'authentification, reportez-vous à la section authentification de votre type de pilote :

- "Authentification NAS ONTAP"
- "Authentification SAN de ONTAP"

Trouvez plus d'informations

- "Documentation Amazon FSX pour NetApp ONTAP"
- "Billet de blog sur Amazon FSX pour NetApp ONTAP"

Intégration d'Amazon FSX pour NetApp ONTAP

Vous pouvez intégrer votre système de fichiers Amazon FSX pour NetApp ONTAP avec Astra Trident pour vous assurer que les clusters Kubernetes exécutés dans Amazon Elastic Kubernetes Service (EKS) peuvent provisionner des volumes persistants de bloc et de fichier sauvegardés par ONTAP.

Avant de commencer

En plus de "Exigences d'Astra Trident", Pour intégrer FSX pour ONTAP avec Astra Trident, vous avez besoin de :

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

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

Exigences supplémentaires pour les volumes SMB

- Cluster Kubernetes avec un nœud de contrôleur Linux et au moins un nœud worker Windows exécutant Windows Server 2019. Astra Trident prend en charge les volumes SMB montés sur des pods qui s'exécutent uniquement sur des nœuds Windows.
- Au moins un secret Astra Trident contenant vos identifiants Active Directory. Pour générer un secret `smbcreds`:

```
kubectl create secret generic smbcreds --from-literal username=user
--from-literal password='password'
```

- Un proxy CSI configuré en tant que service Windows. Pour configurer un `csi-proxy`, voir "GitHub : proxy CSI" ou "GitHub : proxy CSI pour Windows" Pour les nœuds Kubernetes s'exécutant sur Windows.

Intégration des pilotes SAN et NAS de ONTAP

> Si vous configurez la configuration pour les volumes SMB, vous devez lire Préparez-vous au provisionnement des volumes SMB avant de créer le backend.

Étapes

1. Déployez Astra Trident avec l'un des "méthodes de déploiement".

2. Collectez votre nom DNS de la LIF de gestion du SVM. Par exemple, recherchez le sur l'interface de ligne de commandes AWS `DNSName` entrée sous `Endpoints` → `Management` après avoir exécuté la commande suivante :

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

3. Créer et installer des certificats pour "Authentification NAS backend" ou "Authentification SAN backend".

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

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

YAML

```
---
version: 1
storageDriverName: ontap-san
backendName: customBackendName
managementLIF: svm-XXXXXXXXXXXXXXXXX.fs-XXXXXXXXXXXXXXXXX.fsx.us-
east-2.aws.internal
svm: svm01
clientCertificate: ZXR0ZXJwYXB...ICMgJ3BhcGVyc2
clientPrivateKey: vciwKIyAgZG...0cnksIGRlc2NyaX
trustedCACertificate: zcyBbaG...b3Igb3duIGNsYXNz
```

JSON

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

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

- "Configurer un système back-end avec les pilotes NAS ONTAP"
- "Configurer un système back-end avec les pilotes SAN ONTAP"

Résultats

Après déploiement, vous pouvez créer un "classe de stockage, provisionnez un volume et montez le volume dans un pod".

Préparez-vous au provisionnement des volumes SMB

Vous pouvez provisionner des volumes SMB à l'aide de `ontap-nas` conducteur. Avant de terminer Intégration des pilotes SAN et NAS de ONTAP procédez comme suit.

Étapes

1. Création de partages SMB. Vous pouvez créer les partages d'administration SMB de deux manières à l'aide de l' "Console de gestion Microsoft" Dossier partagé snap-in ou à l'aide de l'interface de ligne de commande ONTAP. Pour créer les partages SMB à l'aide de l'interface de ligne de commandes ONTAP :
   a. Si nécessaire, créez la structure du chemin d'accès au répertoire pour le partage.

      Le `vserver cifs share create` commande vérifie le chemin spécifié dans l'option -path lors de la création du partage. Si le chemin spécifié n'existe pas, la commande échoue.

   b. Créer un partage SMB associé au SVM spécifié :

      ```
      vserver cifs share create -vserver vserver_name -share-name
      share_name -path path [-share-properties share_properties,...]
      [other_attributes] [-comment text]
      ```

   c. Vérifiez que le partage a été créé :

      ```
      vserver cifs share show -share-name share_name
      ```

      Reportez-vous à la section "Créez un partage SMB" pour en savoir plus.

2. Lors de la création du back-end, vous devez configurer le suivant pour spécifier les volumes SMB. Pour toutes les options de configuration back-end FSX pour ONTAP, voir "Exemples et options de configuration de FSX pour ONTAP".

Paramètre	Description	Exemple
`smbShare`	Nom du partage SMB créé à l'aide du dossier partagé Microsoft Management Console. Par exemple « smb-share ». **Requis pour les volumes SMB.**	`smb-share`
`nasType`	**Doit être défini sur `smb`.** si elle est nulle, la valeur par défaut est `nfs`.	`smb`

Paramètre	Description	Exemple
`securityStyle`	Style de sécurité pour les nouveaux volumes. **Doit être défini sur `ntfs` ou `mixed` Pour les volumes SMB.**	`ntfs` ou `mixed` Pour les volumes SMB
`unixPermissions`	Mode pour les nouveaux volumes. **Doit rester vide pour les volumes SMB.**	« »

Exemples et options de configuration de FSX pour ONTAP

Découvrez les options de configuration back-end pour Amazon FSX pour ONTAP. Cette section fournit des exemples de configuration back-end.

Options de configuration du back-end

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

Paramètre	Description	Exemple
`version`		Toujours 1
`storageDriverName`	Nom du pilote de stockage	ontap-nas, ontap-nas-économie, ontap-nas-flexgroup, ontap-san », « ontap-san », « ontap-économie san »
`backendName`	Nom personnalisé ou système back-end de stockage	Nom du pilote + "_" + dataLIF
`managementLIF`	Adresse IP d'un cluster ou d'une LIF de gestion SVM pour un basculement MetroCluster transparent, vous devez spécifier une LIF de gestion SVM. Un nom de domaine complet (FQDN) peut être spécifié. Peut être configuré pour utiliser des adresses IPv6 si Astra Trident a été installé à l'aide du `--use-ipv6` drapeau. Les adresses IPv6 doivent être définies entre crochets, telles que [28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555].	« 10.0.0.1 », « [2001:1234:abcd::fefe] »

Paramètre	Description	Exemple
`dataLIF`	Adresse IP de la LIF de protocole. **Pilotes NAS ONTAP**: Nous vous recommandons de spécifier dataLIF. Si elle n'est pas fournie, Astra Trident extrait les LIF de données du SVM. Vous pouvez spécifier un nom de domaine complet (FQDN) à utiliser pour les opérations de montage NFS, permettant de créer un DNS Round-Robin pour équilibrer la charge sur plusieurs LIF de données. Peut être modifié après le réglage initial. Reportez-vous à la section . **Pilotes SAN ONTAP** : ne pas spécifier pour iSCSI. Astra Trident utilise le mappage de LUN sélectif de ONTAP pour découvrir les LIFs iSCI nécessaires pour établir une session multi-chemins. Un avertissement est généré si dataLIF est explicitement défini. Peut être configuré pour utiliser des adresses IPv6 si Astra Trident a été installé à l'aide du `--use-ipv6` drapeau. Les adresses IPv6 doivent être définies entre crochets, telles que [28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555].	
`autoExportPolicy`	Activer la création et la mise à jour automatiques des règles d'exportation [booléennes]. À l'aide du `autoExportPolicy` et `autoExportCIDRs` Avec Astra Trident, il peut gérer automatiquement les règles d'exportation.	« faux »
`autoExportCIDRs`	Liste des CIDR pour filtrer les adresses IP du nœud Kubernetes par rapport à quand `autoExportPolicy` est activé. À l'aide du `autoExportPolicy` et `autoExportCIDRs` Avec Astra Trident, il peut gérer automatiquement les règles d'exportation.	« [« 0.0.0.0/0 », «:/0 »] »
`labels`	Ensemble d'étiquettes arbitraires au format JSON à appliquer aux volumes	« »

Paramètre	Description	Exemple
`clientCertificate`	Valeur encodée en Base64 du certificat client. Utilisé pour l'authentification par certificat	« »
`clientPrivateKey`	Valeur encodée en Base64 de la clé privée du client. Utilisé pour l'authentification par certificat	« »
`trustedCACertificate`	Valeur encodée en Base64 du certificat CA de confiance. Facultatif. Utilisé pour l'authentification basée sur des certificats.	« »
`username`	Nom d'utilisateur pour la connexion au cluster ou au SVM. Utilisé pour l'authentification basée sur les identifiants. Par exemple, vsadmin.	
`password`	Mot de passe pour se connecter au cluster ou au SVM. Utilisé pour l'authentification basée sur les identifiants.	
`svm`	Serveur virtuel de stockage à utiliser	Dérivé si une LIF de gestion SVM est spécifiée.
`igroupName`	Nom du groupe initiateur à utiliser pour les volumes SAN. Reportez-vous à la section .	Trident-<backend-UUID>
`storagePrefix`	Préfixe utilisé pour le provisionnement des nouveaux volumes dans la SVM. Ne peut pas être modifié après sa création. Pour mettre à jour ce paramètre, vous devez créer un nouveau backend.	trident
`limitAggregateUsage`	**Ne pas spécifier pour Amazon FSX pour NetApp ONTAP** `fsxadmin` et `vsadmin` Ne contiennent pas les autorisations requises pour récupérer l'utilisation d'agrégats et le limiter à l'aide d'Astra Trident.	Ne pas utiliser.
`limitVolumeSize`	Echec du provisionnement si la taille du volume demandé est supérieure à cette valeur. Restreint également la taille maximale des volumes qu'il gère pour les qtrees et les LUN, et la `qtreesPerFlexvol` L'option permet de personnaliser le nombre maximal de qtree par FlexVol.	« » (non appliqué par défaut)

Paramètre	Description	Exemple
`lunsPerFlexvol`	Le nombre maximal de LUN par FlexVol doit être compris dans la plage [50, 200]. SAN uniquement.	« 100 »
`debugTraceFlags`	Indicateurs de débogage à utiliser lors du dépannage. Par exemple, {“api”:false, “méthode”:true} ne pas utiliser `debugTraceFlags` à moins que vous ne soyez en mesure de dépanner et que vous ayez besoin d’un vidage détaillé des journaux.	nul
`nfsMountOptions`	Liste des options de montage NFS séparée par des virgules. Les options de montage des volumes Kubernetes persistants sont généralement spécifiées dans les classes de stockage, mais si aucune option de montage n’est spécifiée dans une classe de stockage, Astra Trident utilisera les options de montage spécifiées dans le fichier de configuration du système back-end. Si aucune option de montage n’est spécifiée dans la classe de stockage ou le fichier de configuration, Astra Trident ne définit aucune option de montage sur un volume persistant associé.	« »
`nasType`	Configurez la création de volumes NFS ou SMB. Les options sont `nfs`, `smb`, ou nul. **Doit être défini sur `smb` Pour les volumes SMB.** la valeur NULL est définie par défaut sur les volumes NFS.	nfs
`qtreesPerFlexvol`	Nombre maximal de qtrees par FlexVol, qui doit être compris dans la plage [50, 300]	« 200 »
`smbShare`	Nom du partage SMB créé à l’aide du dossier partagé Microsoft Management Console. **Requis pour les volumes SMB.**	« partage smb »

Paramètre	Description	Exemple
`useREST`	Paramètre booléen pour utiliser les API REST de ONTAP. **Aperçu technique** `useREST` est fourni sous forme d'aperçu technique ** qui est recommandé pour les environnements de test et non pour les charges de travail de production. Lorsqu'il est réglé sur `true`, Astra Trident va utiliser les API REST de ONTAP pour communiquer avec le système back-end. Cette fonctionnalité requiert ONTAP 9.11.1 et versions ultérieures. En outre, le rôle de connexion ONTAP utilisé doit avoir accès au `ontap` client supplémentaire. Ceci est satisfait par le pré-défini `vsadmin` et `cluster-admin` rôles.	« faux »

Détails sur `igroupName`

`igroupName` Peut être défini sur un groupe initiateur déjà créé sur le cluster ONTAP. Si non spécifié, Astra Trident crée automatiquement un groupe initiateur nommé `trident-<backend-UUID>`.

Si vous disposez d'un nom de groupe prédéfini, nous vous recommandons d'utiliser un groupe initiateur par cluster Kubernetes si le SVM doit être partagé entre les environnements. Cela est nécessaire pour qu'Astra Trident conserve automatiquement les ajouts et suppressions d'IQN.

- `igroupName` Peut être mis à jour afin de désigner un nouveau groupe initiateur créé et géré sur la SVM en dehors d'Astra Trident.
- `igroupName` peut être omis. Dans ce cas, Astra Trident crée et gère un groupe initiateur nommé `trident-<backend-UUID>` automatiquement.

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

Mise à jour `dataLIF` **après la configuration initiale**

Vous pouvez modifier la LIF de données après la configuration initiale en exécutant la commande suivante pour fournir le nouveau fichier JSON back-end avec la LIF de données mise à jour.

```
tridentctl update backend <backend-name> -f <path-to-backend-json-file-
with-updated-dataLIF>
```

Si des demandes de volume persistant sont associées à un ou plusieurs pods, tous les pods correspondants doivent être arrêtés, puis réintégrés dans le but de permettre la nouvelle LIF de données d'être effective.

Options de configuration back-end pour les volumes de provisionnement

Vous pouvez contrôler le provisionnement par défaut à l'aide de ces options dans `defaults` section de la configuration. Pour un exemple, voir les exemples de configuration ci-dessous.

Paramètre	Description	Valeur par défaut
`spaceAllocation`	Allocation d'espace pour les LUN	« vrai »
`spaceReserve`	Mode de réservation d'espace ; "none" (fin) ou "volume" (épais)	« aucun »
`snapshotPolicy`	Règle Snapshot à utiliser	« aucun »
`qosPolicy`	QoS policy group à affecter pour les volumes créés. Choisissez une de qosPolicy ou adaptiveQosPolicy par pool de stockage ou back-end. Avec Astra Trident, les groupes de règles de QoS doivent être utilisés avec ONTAP 9.8 ou version ultérieure. Nous recommandons l'utilisation d'un groupe de règles de qualité de service non partagé et nous assurer que le groupe de règles est appliqué à chaque composant individuellement. Un groupe de règles de QoS partagé appliquera le plafond du débit total de toutes les charges de travail.	« »
`adaptiveQosPolicy`	Groupe de règles de QoS adaptative à attribuer aux volumes créés. Choisissez une de qosPolicy ou adaptiveQosPolicy par pool de stockage ou back-end. Non pris en charge par l'économie ontap-nas.	« »
`snapshotReserve`	Pourcentage du volume réservé pour les instantanés "0"	Si `snapshotPolicy` est « aucun », sinon « »
`splitOnClone`	Séparer un clone de son parent lors de sa création	« faux »

Paramètre	Description	Valeur par défaut
`encryption`	Activez NetApp Volume Encryption (NVE) sur le nouveau volume. La valeur par défaut est `false`. Pour utiliser cette option, NVE doit être sous licence et activé sur le cluster. Si NAE est activé sur le back-end, tous les volumes provisionnés dans Astra Trident seront activés par NAE. Pour plus d'informations, se reporter à : "Fonctionnement d'Astra Trident avec NVE et NAE".	« faux »
`luksEncryption`	Activez le cryptage LUKS. Reportez-vous à la section "Utiliser la configuration de clé unifiée Linux (LUKS)". SAN uniquement.	« »
`tieringPolicy`	La stratégie de hiérarchisation à utiliser « none »	Snapshot uniquement pour une configuration SVM-DR pré-ONTAP 9.5
`unixPermissions`	Mode pour les nouveaux volumes. **Laisser vide pour les volumes SMB.**	« »
`securityStyle`	Style de sécurité pour les nouveaux volumes. Prise en charge de NFS `mixed` et `unix` styles de sécurité. SMB prend en charge `mixed` et `ntfs` styles de sécurité.	NFS par défaut est `unix`. SMB par défaut est `ntfs`.

Exemple

À l'aide de `nasType`, `node-stage-secret-name`, et `node-stage-secret-namespace`, Vous pouvez spécifier un volume SMB et fournir les informations d'identification Active Directory requises. Les volumes SMB sont pris en charge à l'aide de `ontap-nas` conducteur uniquement.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: nas-smb-sc
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-nas"
  trident.netapp.io/nasType: "smb"
  csi.storage.k8s.io/node-stage-secret-name: "smbcreds"
  csi.storage.k8s.io/node-stage-secret-namespace: "default"
```

Création de systèmes back-end avec kubectl

Un système back-end définit la relation entre Astra Trident et un système de stockage. Il explique à Astra Trident comment communiquer avec ce système de stockage et comment Astra Trident doit provisionner des volumes à partir de celui-ci. Après l'installation d'Astra Trident, l'étape suivante consiste à créer un système back-end. Le `TridentBackendConfig` La définition de ressource personnalisée (CRD) vous permet de créer et de gérer des systèmes back-end Trident directement via l'interface Kubernetes. Vous pouvez le faire en utilisant `kubectl` Ou l'outil CLI équivalent pour votre distribution Kubernetes.

`TridentBackendConfig`

`TridentBackendConfig` (`tbc`, `tbconfig`, `tbackendconfig`) Est un CRD au rythme de noms qui vous permet de gérer les systèmes back-end Astra Trident à l'aide de `kubectl`. Avec Kubernetes et les administrateurs de stockage, il est désormais possible de créer et de gérer des systèmes back-end directement via la CLI Kubernetes sans avoir besoin d'un utilitaire de ligne de commande dédié (`tridentctl`).

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

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

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

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

Reportez-vous à l'exemple suivant pour connaître le format du `TridentBackendConfig` CR :

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-san
spec:
  version: 1
  backendName: ontap-san-backend
  storageDriverName: ontap-san
  managementLIF: 10.0.0.1
  dataLIF: 10.0.0.2
  svm: trident_svm
  credentials:
    name: backend-tbc-ontap-san-secret
```

Vous pouvez également consulter les exemples de la "programme d'installation trident" répertoire des

exemples de configuration pour la plate-forme/le service de stockage souhaité.

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

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

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

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

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

Présentation des étapes

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

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

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

Étape 1 : créez un code secret Kubernetes

Créez un secret qui contient les informations d'identification d'accès pour le back-end. Ce point est unique à chaque service/plateforme de stockage. Voici un exemple :

```
kubectl -n trident create -f backend-tbc-ontap-san-secret.yaml
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-ontap-san-secret
type: Opaque
stringData:
  username: cluster-admin
  password: password
```

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

Description des champs secrets de la plate-forme de stockage	Secret	Description des champs
Azure NetApp Files	ID client	ID client d'un enregistrement d'application
Cloud Volumes Service pour GCP	id_clé_privée	ID de la clé privée. Partie de la clé API pour le compte de service GCP avec le rôle d'administrateur CVS
Cloud Volumes Service pour GCP	clé_privée	Clé privée. Partie de la clé API pour le compte de service GCP avec le rôle d'administrateur CVS
Element (NetApp HCI/SolidFire)	Point final	MVIP pour le cluster SolidFire avec les identifiants de locataire
ONTAP	nom d'utilisateur	Nom d'utilisateur pour la connexion au cluster/SVM. Utilisé pour l'authentification basée sur les identifiants
ONTAP	mot de passe	Mot de passe pour la connexion au cluster/SVM. Utilisé pour l'authentification basée sur les identifiants
ONTAP	ClientPrivateKey	Valeur encodée en Base64 de la clé privée du client. Utilisé pour l'authentification basée sur des certificats

Description des champs secrets de la plate-forme de stockage	Secret	Description des champs
ONTAP	ChapUsername	Nom d'utilisateur entrant. Requis si useCHAP=vrai. Pour `ontap-san` et `ontap-san-economy`
ONTAP	Chapeau InitiatorSecret	Secret de l'initiateur CHAP. Requis si useCHAP=vrai. Pour `ontap-san` et `ontap-san-economy`
ONTAP	ChapTargetUsername	Nom d'utilisateur cible. Requis si useCHAP=vrai. Pour `ontap-san` et `ontap-san-economy`
ONTAP	ChapTargetInitiatorSecret	Secret de l'initiateur cible CHAP. Requis si useCHAP=vrai. Pour `ontap-san` et `ontap-san-economy`

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

Étape 2 : créez le `TridentBackendConfig` CR

Vous êtes maintenant prêt à créer votre `TridentBackendConfig` CR. Dans cet exemple, un back-end qui utilise le `ontap-san` le pilote est créé à l'aide du `TridentBackendConfig` objet illustré ci-dessous :

```
kubectl -n trident create -f backend-tbc-ontap-san.yaml
```

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-san
spec:
  version: 1
  backendName: ontap-san-backend
  storageDriverName: ontap-san
  managementLIF: 10.0.0.1
  dataLIF: 10.0.0.2
  svm: trident_svm
  credentials:
    name: backend-tbc-ontap-san-secret
```

Étape 3 : vérifier l'état du `TridentBackendConfig` CR

Maintenant que vous avez créé le `TridentBackendConfig` CR, vous pouvez vérifier l'état. Voir l'exemple suivant :

```
kubectl -n trident get tbc backend-tbc-ontap-san
NAME                       BACKEND NAME          BACKEND UUID
PHASE   STATUS
backend-tbc-ontap-san   ontap-san-backend     8d24fce7-6f60-4d4a-8ef6-
bab2699e6ab8   Bound   Success
```

Un back-end a été créé avec succès et lié au `TridentBackendConfig` CR.

La phase peut prendre l'une des valeurs suivantes :

- `Bound`: Le `TridentBackendConfig` La demande de modification est associée à un back-end, et ce backend contient `configRef` réglez sur `TridentBackendConfig` L'uid de CR.
- `Unbound`: Représenté en utilisant `""`. Le `TridentBackendConfig` l'objet n'est pas lié à un back-end. Tout nouveau `TridentBackendConfig` Les CRS sont dans cette phase par défaut. Une fois la phase modifiée, elle ne peut plus revenir à Unbound.
- `Deleting`: Le `TridentBackendConfig` CR `deletionPolicy` a été configuré pour supprimer. Lorsque le `TridentBackendConfig` La demande de modification est supprimée, elle passe à l'état Suppression.
  - Si aucune demande de volume persistant n'existe sur le back-end, supprimez le `TridentBackendConfig` Il en résultera la suppression du système back-end et du système Astra Trident `TridentBackendConfig` CR.
  - Si un ou plusieurs ESV sont présents sur le back-end, il passe à l'état de suppression. Le `TridentBackendConfig` La CR entre ensuite la phase de suppression. Le back-end et `TridentBackendConfig` Sont supprimés uniquement après la suppression de tous les ESV.
- `Lost`: Le back-end associé à l' `TridentBackendConfig` Le CR a été accidentellement ou délibérément supprimé et le `TridentBackendConfig` La CR a toujours une référence au back-end supprimé. Le `TridentBackendConfig` La CR peut toujours être supprimée, quel que soit le `deletionPolicy` valeur.
- `Unknown`: Astra Trident n'est pas en mesure de déterminer l'état ou l'existence du back-end associé au `TridentBackendConfig` CR. Par exemple, si le serveur d'API ne répond pas ou si `tridentbackends.trident.netapp.io` CRD manquant. Cela peut nécessiter l'intervention de l'utilisateur.

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

(Facultatif) étape 4 : pour plus de détails

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

```
kubectl -n trident get tbc backend-tbc-ontap-san -o wide
```

```
NAME                      BACKEND NAME        BACKEND UUID
PHASE   STATUS    STORAGE DRIVER   DELETION POLICY
backend-tbc-ontap-san   ontap-san-backend   8d24fce7-6f60-4d4a-8ef6-
bab2699e6ab8   Bound   Success   ontap-san        delete
```

En outre, vous pouvez également obtenir un vidage YAML/JSON de `TridentBackendConfig`.

```
kubectl -n trident get tbc backend-tbc-ontap-san -o yaml
```

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  creationTimestamp: "2021-04-21T20:45:11Z"
  finalizers:
  - trident.netapp.io
  generation: 1
  name: backend-tbc-ontap-san
  namespace: trident
  resourceVersion: "947143"
  uid: 35b9d777-109f-43d5-8077-c74a4559d09c
spec:
  backendName: ontap-san-backend
  credentials:
    name: backend-tbc-ontap-san-secret
  managementLIF: 10.0.0.1
  dataLIF: 10.0.0.2
  storageDriverName: ontap-san
  svm: trident_svm
  version: 1
status:
  backendInfo:
    backendName: ontap-san-backend
    backendUUID: 8d24fce7-6f60-4d4a-8ef6-bab2699e6ab8
  deletionPolicy: delete
  lastOperationStatus: Success
  message: Backend 'ontap-san-backend' created
  phase: Bound
```

`backendInfo` contient le `backendName` et le `backendUUID` du back-end créé en réponse à `TridentBackendConfig` CR. Le `lastOperationStatus` champ représente l'état de la dernière opération du `TridentBackendConfig` CR, qui peut être déclenché par l'utilisateur (par exemple, l'utilisateur a modifié quelque chose dans `spec`) Ou déclenché par Astra Trident (par exemple lors du redémarrage d'Astra Trident). Il peut être réussi ou échoué. `phase` représente l'état de la relation entre `TridentBackendConfig` CR et le

backend. Dans l'exemple ci-dessus, `phase` A la valeur limitée, ce qui signifie que le `TridentBackendConfig` CR est associé au back-end.

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

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

Effectuer la gestion back-end avec kubectl

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

Supprimer un back-end

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

Exécutez la commande suivante :

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

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

Affichez les systèmes back-end existants

Exécutez la commande suivante :

```
kubectl get tbc -n trident
```

Vous pouvez également exécuter `tridentctl get backend -n trident` ou `tridentctl get backend -o yaml -n trident` pour obtenir une liste de tous les systèmes back-end existants, Cette liste comprend également les systèmes back-end créés avec `tridentctl`.

Mettre à jour un back-end

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

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

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

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

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

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

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

En cas d'échec d'une mise à jour du back-end, le système back-end continue de rester dans sa dernière configuration connue. Vous pouvez afficher les journaux pour déterminer la cause en cours d'exécution `kubectl get tbc <tbc-name> -o yaml -n trident` ou `kubectl describe tbc <tbc-name> -n trident`.

Après avoir identifié et corrigé le problème avec le fichier de configuration, vous pouvez relancer la commande update.

Gestion back-end avec tridentctl

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

Créer un back-end

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

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

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

```
tridentctl logs -n trident
```

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

Supprimer un back-end

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

1. Récupérer le nom du système back-end :

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

2. Supprimer le backend :

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

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

Affichez les systèmes back-end existants

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

- Pour obtenir un récapitulatif, exécutez la commande suivante :

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

- Pour obtenir tous les détails, exécutez la commande suivante :

```
tridentctl get backend -o json -n trident
```

Mettre à jour un back-end

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

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

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

```
tridentctl logs -n trident
```

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

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

Voici un exemple de questions que vous pouvez répondre avec le fichier JSON `tridentctl` sorties des

objets back-end. Ceci utilise le `jq` utilitaire que vous devez installer.

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

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

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

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

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

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

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

Cela s'applique aux scénarios suivants :

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

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

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

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

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

Étape 0 : identifier le back-end

Pour créer un `TridentBackendConfig` qui se lie à un back-end existant, vous devez obtenir la configuration back-end. Dans cet exemple, supposons qu'un back-end a été créé à l'aide de la définition JSON suivante :

```
tridentctl get backend ontap-nas-backend -n trident
+---------------------+----------------
+--------------------------------------+--------+---------+
|          NAME       | STORAGE DRIVER |                 UUID
| STATE  | VOLUMES |
+---------------------+----------------
+--------------------------------------+--------+---------+
| ontap-nas-backend   | ontap-nas      | 52f2eb10-e4c6-4160-99fc-
96b3be5ab5d7 | online |      25 |
+---------------------+----------------
+--------------------------------------+--------+---------+

cat ontap-nas-backend.json

{
    "version": 1,
    "storageDriverName": "ontap-nas",
    "managementLIF": "10.10.10.1",
    "dataLIF": "10.10.10.2",
    "backendName": "ontap-nas-backend",
    "svm": "trident_svm",
    "username": "cluster-admin",
    "password": "admin-password",

    "defaults": {
        "spaceReserve": "none",
        "encryption": "false"
    },
    "labels":{"store":"nas_store"},
    "region": "us_east_1",
    "storage": [
        {
            "labels":{"app":"msoffice", "cost":"100"},
            "zone":"us_east_1a",
            "defaults": {
                "spaceReserve": "volume",
                "encryption": "true",
                "unixPermissions": "0755"
            }
        },
        {
```

```
            "labels":{"app":"mysqldb", "cost":"25"},
            "zone":"us_east_1d",
            "defaults": {
                "spaceReserve": "volume",
                "encryption": "false",
                "unixPermissions": "0775"
            }
        }
    ]
}
```

Étape 1 : créez un code secret Kubernetes

Créez un secret qui contient les informations d'identification du back-end, comme indiqué dans cet exemple :

```
cat tbc-ontap-nas-backend-secret.yaml

apiVersion: v1
kind: Secret
metadata:
  name: ontap-nas-backend-secret
type: Opaque
stringData:
  username: cluster-admin
  password: admin-password

kubectl create -f tbc-ontap-nas-backend-secret.yaml -n trident
secret/backend-tbc-ontap-san-secret created
```

Étape 2 : créer un `TridentBackendConfig` CR

L'étape suivante consiste à créer un `TridentBackendConfig` CR qui se lie automatiquement au pré-existant `ontap-nas-backend` (comme dans cet exemple). Assurez-vous que les exigences suivantes sont respectées :

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

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

```
cat backend-tbc-ontap-nas.yaml
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: tbc-ontap-nas-backend
spec:
  version: 1
  storageDriverName: ontap-nas
  managementLIF: 10.10.10.1
  dataLIF: 10.10.10.2
  backendName: ontap-nas-backend
  svm: trident_svm
  credentials:
    name: mysecret
  defaults:
    spaceReserve: none
    encryption: 'false'
  labels:
    store: nas_store
  region: us_east_1
  storage:
  - labels:
      app: msoffice
      cost: '100'
    zone: us_east_1a
    defaults:
      spaceReserve: volume
      encryption: 'true'
      unixPermissions: '0755'
  - labels:
      app: mysqldb
      cost: '25'
    zone: us_east_1d
    defaults:
      spaceReserve: volume
      encryption: 'false'
      unixPermissions: '0775'

kubectl create -f backend-tbc-ontap-nas.yaml -n trident
tridentbackendconfig.trident.netapp.io/tbc-ontap-nas-backend created
```

Étape 3 : vérifier l'état du `TridentBackendConfig` CR

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

```
kubectl -n trident get tbc tbc-ontap-nas-backend -n trident
NAME                  BACKEND NAME          BACKEND UUID
PHASE   STATUS
tbc-ontap-nas-backend  ontap-nas-backend     52f2eb10-e4c6-4160-99fc-
96b3be5ab5d7   Bound   Success

#confirm that no new backends were created (i.e., TridentBackendConfig did
not end up creating a new backend)
tridentctl get backend -n trident
+---------------------+----------------
+--------------------------------------+--------+---------+
|          NAME       | STORAGE DRIVER |                 UUID
| STATE  | VOLUMES |
+---------------------+----------------
+--------------------------------------+--------+---------+
| ontap-nas-backend   | ontap-nas      | 52f2eb10-e4c6-4160-99fc-
96b3be5ab5d7 | online |      25 |
+---------------------+----------------
+--------------------------------------+--------+---------+
```

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

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

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

Étape 0 : identifier le back-end

Par exemple, supposons que le back-end suivant a été créé à l'aide de `TridentBackendConfig`:

```
kubectl get tbc backend-tbc-ontap-san -n trident -o wide
NAME                    BACKEND NAME        BACKEND UUID
PHASE   STATUS    STORAGE DRIVER   DELETION POLICY
backend-tbc-ontap-san   ontap-san-backend   81abcb27-ea63-49bb-b606-
0a5315ac5f82   Bound   Success   ontap-san        delete

tridentctl get backend ontap-san-backend -n trident
+-------------------+----------------
+--------------------------------------+--------+---------+
|       NAME        | STORAGE DRIVER |                 UUID
| STATE  | VOLUMES |
+-------------------+----------------
+--------------------------------------+--------+---------+
| ontap-san-backend | ontap-san      | 81abcb27-ea63-49bb-b606-
0a5315ac5f82 | online |      33 |
+-------------------+----------------
+--------------------------------------+--------+---------+
```

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

Étape 1 : confirmer `deletionPolicy` **est défini sur** `retain`

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

```
kubectl get tbc backend-tbc-ontap-san -n trident -o wide
NAME                    BACKEND NAME        BACKEND UUID
PHASE   STATUS    STORAGE DRIVER   DELETION POLICY
backend-tbc-ontap-san   ontap-san-backend   81abcb27-ea63-49bb-b606-
0a5315ac5f82   Bound   Success   ontap-san        delete

# Patch value of deletionPolicy to retain
kubectl patch tbc backend-tbc-ontap-san --type=merge -p
'{"spec":{"deletionPolicy":"retain"}}' -n trident
tridentbackendconfig.trident.netapp.io/backend-tbc-ontap-san patched

#Confirm the value of deletionPolicy
kubectl get tbc backend-tbc-ontap-san -n trident -o wide
NAME                    BACKEND NAME        BACKEND UUID
PHASE   STATUS    STORAGE DRIVER   DELETION POLICY
backend-tbc-ontap-san   ontap-san-backend   81abcb27-ea63-49bb-b606-
0a5315ac5f82   Bound   Success   ontap-san        retain
```

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

Étape 2 : supprimez le `TridentBackendConfig` CR

La dernière étape consiste à supprimer le `TridentBackendConfig` CR. Après avoir confirmé le `deletionPolicy` est défini sur `retain`, vous pouvez poursuivre la suppression :

```
kubectl delete tbc backend-tbc-ontap-san -n trident
tridentbackendconfig.trident.netapp.io "backend-tbc-ontap-san" deleted

tridentctl get backend ontap-san-backend -n trident
+------------------+----------------
+--------------------------------------+--------+---------+
|       NAME        | STORAGE DRIVER |                 UUID
| STATE   | VOLUMES |
+------------------+----------------
+--------------------------------------+--------+---------+
| ontap-san-backend | ontap-san      | 81abcb27-ea63-49bb-b606-
0a5315ac5f82 | online |      33 |
+------------------+----------------
+--------------------------------------+--------+---------+
```

Lors de la suppression du `TridentBackendConfig` Objet : Astra Trident la supprime simplement sans le système back-end.

Gérer les classes de stockage

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

Concevez une classe de stockage

Voir "ici" pour plus d'informations sur les classes de stockage et leur configuration.

Créer une classe de stockage

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

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

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

Supprimer une classe de stockage

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

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

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

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

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

Afficher les classes de stockage existantes

- Pour afficher les classes de stockage Kubernetes existantes, exécutez la commande suivante :

```
kubectl get storageclass
```

- Pour afficher les détails de la classe de stockage Kubernetes, exécutez la commande suivante :

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

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

```
tridentctl get storageclass
```

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

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

Définir une classe de stockage par défaut

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

- Définissez une classe de stockage par défaut en définissant l'annotation `storageclass.kubernetes.io/is-default-class` vrai dans la définition de classe de stockage. Selon la spécification, toute autre valeur ou absence de l'annotation est interprétée comme fausse.
- Vous pouvez configurer une classe de stockage existante comme classe de stockage par défaut à l'aide de la commande suivante :

```
kubectl patch storageclass <storage-class-name> -p '{"metadata":
{"annotations":{"storageclass.kubernetes.io/is-default-class":"true"}}}'
```

- De même, vous pouvez supprimer l'annotation de classe de stockage par défaut à l'aide de la commande suivante :

```
kubectl patch storageclass <storage-class-name> -p '{"metadata":
{"annotations":{"storageclass.kubernetes.io/is-default-class":"false"}}}'
```

Il existe également des exemples dans le bundle du programme d'installation de Trident qui incluent cette annotation.

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

Identifier le système back-end pour une classe de stockage

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

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

Réaliser des opérations de volume

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

- "Utiliser la topologie CSI"
- "Travailler avec des instantanés"
- "Développement des volumes"
- "Importer des volumes"

Utiliser la topologie CSI

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

En savoir plus sur la fonction de topologie CSI "ici".

Kubernetes propose deux modes de liaison de volumes :

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

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

Ce dont vous avez besoin

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

- Un cluster Kubernetes exécutant un "Version Kubernetes prise en charge"

```
kubectl version
Client Version: version.Info{Major:"1", Minor:"19",
GitVersion:"v1.19.3",
GitCommit:"1e11e4a2108024935ecfcb2912226cedeafd99df",
GitTreeState:"clean", BuildDate:"2020-10-14T12:50:19Z",
GoVersion:"go1.15.2", Compiler:"gc", Platform:"linux/amd64"}
Server Version: version.Info{Major:"1", Minor:"19",
GitVersion:"v1.19.3",
GitCommit:"1e11e4a2108024935ecfcb2912226cedeafd99df",
GitTreeState:"clean", BuildDate:"2020-10-14T12:41:49Z",
GoVersion:"go1.15.2", Compiler:"gc", Platform:"linux/amd64"}
```

- Les nœuds du cluster doivent avoir des étiquettes qui permettent la prise en charge de la topologie (`topology.kubernetes.io/region` et `topology.kubernetes.io/zone`). Ces étiquettes **doivent être présentes sur les nœuds du cluster** avant d'installer Astra Trident pour qu'Astra Trident soit compatible avec la topologie.

```
kubectl get nodes -o=jsonpath='{range .items[*]}[{.metadata.name},
{.metadata.labels}]{"\n"}{end}' | grep --color "topology.kubernetes.io"
[node1,
{"beta.kubernetes.io/arch":"amd64","beta.kubernetes.io/os":"linux","kube
rnetes.io/arch":"amd64","kubernetes.io/hostname":"node1","kubernetes.io/
os":"linux","node-
role.kubernetes.io/master":"","topology.kubernetes.io/region":"us-
east1","topology.kubernetes.io/zone":"us-east1-a"}]
[node2,
{"beta.kubernetes.io/arch":"amd64","beta.kubernetes.io/os":"linux","kube
rnetes.io/arch":"amd64","kubernetes.io/hostname":"node2","kubernetes.io/
os":"linux","node-
role.kubernetes.io/worker":"","topology.kubernetes.io/region":"us-
east1","topology.kubernetes.io/zone":"us-east1-b"}]
[node3,
{"beta.kubernetes.io/arch":"amd64","beta.kubernetes.io/os":"linux","kube
rnetes.io/arch":"amd64","kubernetes.io/hostname":"node3","kubernetes.io/
os":"linux","node-
role.kubernetes.io/worker":"","topology.kubernetes.io/region":"us-
east1","topology.kubernetes.io/zone":"us-east1-c"}]
```

Étape 1 : création d'un back-end conscient de la topologie

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

Voici un exemple de définition de back-end :

YAML

```
---
version: 1
storageDriverName: ontap-san
backendName: san-backend-us-east1
managementLIF: 192.168.27.5
svm: iscsi_svm
username: admin
password: password
supportedTopologies:
- topology.kubernetes.io/region: us-east1
  topology.kubernetes.io/zone: us-east1-a
- topology.kubernetes.io/region: us-east1
  topology.kubernetes.io/zone: us-east1-b
```

JSON

```
{
 "version": 1,
 "storageDriverName": "ontap-san",
 "backendName": "san-backend-us-east1",
 "managementLIF": "192.168.27.5",
 "svm": "iscsi_svm",
 "username": "admin",
 "password": "password",
 "supportedTopologies": [
{"topology.kubernetes.io/region": "us-east1",
"topology.kubernetes.io/zone": "us-east1-a"},
{"topology.kubernetes.io/region": "us-east1",
"topology.kubernetes.io/zone": "us-east1-b"}
]
}
```

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

Vous pouvez définir `supportedTopologies` par pool de stockage également. Voir l'exemple suivant :

```
---
version: 1
storageDriverName: ontap-nas
backendName: nas-backend-us-central1
managementLIF: 172.16.238.5
svm: nfs_svm
username: admin
password: password
supportedTopologies:
- topology.kubernetes.io/region: us-central1
  topology.kubernetes.io/zone: us-central1-a
- topology.kubernetes.io/region: us-central1
  topology.kubernetes.io/zone: us-central1-b
storage:
- labels:
    workload: production
  region: Iowa-DC
  zone: Iowa-DC-A
  supportedTopologies:
  - topology.kubernetes.io/region: us-central1
    topology.kubernetes.io/zone: us-central1-a
- labels:
    workload: dev
  region: Iowa-DC
  zone: Iowa-DC-B
  supportedTopologies:
  - topology.kubernetes.io/region: us-central1
    topology.kubernetes.io/zone: us-central1-b
```

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

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

Les classes de stockage peuvent être définies en fonction des labels de topologie fournis aux nœuds du cluster, et contenir des informations de topologie. Cela déterminera les pools de stockage qui servent de candidats aux demandes de volume persistant faites et le sous-ensemble de nœuds qui peuvent utiliser les volumes provisionnés par Trident.

Voir l'exemple suivant :

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
name: netapp-san-us-east1
provisioner: csi.trident.netapp.io
volumeBindingMode: WaitForFirstConsumer
allowedTopologies:
- matchLabelExpressions:
- key: topology.kubernetes.io/zone
  values:
  - us-east1-a
  - us-east1-b
- key: topology.kubernetes.io/region
  values:
  - us-east1
parameters:
  fsType: "ext4"
```

Dans la définition de classe de stockage décrite ci-dessus, `volumeBindingMode` est défini sur `WaitForFirstConsumer`. Les demandes de volume persistant demandées pour cette classe de stockage ne seront pas traitées tant qu'elles ne seront pas référencées dans un pod. Et, `allowedTopologies` fournit les zones et la région à utiliser. Le `netapp-san-us-east1` StorageClass crée des ESV sur le `san-backend-us-east1` système back-end défini ci-dessus.

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

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

Voir l'exemple `spec` ci-dessous :

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

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

```
kubectl create -f pvc.yaml
persistentvolumeclaim/pvc-san created
kubectl get pvc
NAME      STATUS    VOLUME   CAPACITY   ACCESS MODES   STORAGECLASS
AGE
pvc-san   Pending                                      netapp-san-us-east1
2s
kubectl describe pvc
Name:          pvc-san
Namespace:     default
StorageClass:  netapp-san-us-east1
Status:        Pending
Volume:
Labels:        <none>
Annotations:   <none>
Finalizers:    [kubernetes.io/pvc-protection]
Capacity:
Access Modes:
VolumeMode:    Filesystem
Mounted By:    <none>
Events:
  Type    Reason                Age   From                         Message
  ----    ------                ----  ----                         -------
  Normal  WaitForFirstConsumer  6s    persistentvolume-controller  waiting
for first consumer to be created before binding
```

Pour que Trident puisse créer un volume et le lier à la demande de volume persistant, utilisez la demande de volume persistant dans un pod. Voir l'exemple suivant :

```
apiVersion: v1
kind: Pod
metadata:
  name: app-pod-1
spec:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: topology.kubernetes.io/region
            operator: In
            values:
            - us-east1
      preferredDuringSchedulingIgnoredDuringExecution:
      - weight: 1
        preference:
          matchExpressions:
          - key: topology.kubernetes.io/zone
            operator: In
            values:
            - us-east1-a
            - us-east1-b
  securityContext:
    runAsUser: 1000
    runAsGroup: 3000
    fsGroup: 2000
  volumes:
  - name: vol1
    persistentVolumeClaim:
      claimName: pvc-san
  containers:
  - name: sec-ctx-demo
    image: busybox
    command: [ "sh", "-c", "sleep 1h" ]
    volumeMounts:
    - name: vol1
      mountPath: /data/demo
    securityContext:
      allowPrivilegeEscalation: false
```

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

Voir le résultat suivant :

```
kubectl get pods -o wide
NAME        READY   STATUS    RESTARTS   AGE   IP               NODE
NOMINATED NODE   READINESS GATES
app-pod-1   1/1     Running   0          19s   192.168.25.131   node2
<none>           <none>
kubectl get pvc -o wide
NAME      STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS          AGE   VOLUMEMODE
pvc-san   Bound    pvc-ecb1e1a0-840c-463b-8b65-b3d033e2e62b   300Mi
RWO            netapp-san-us-east1   48s   Filesystem
```

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

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

Trouvez plus d'informations

- "Gestion des ressources pour les conteneurs"
- "Outil de sélection de nœud"
- "Affinité et anti-affinité"
- "Teintes et tolérances"

Travailler avec des instantanés

Vous pouvez créer des copies Snapshot de volume Kubernetes VolumeSnapshot (copie de volume) de volumes persistants pour conserver des copies instantanées de volumes Trident. Vous pouvez également créer un nouveau volume, également appelé *clone*, à partir d'un snapshot de volume existant. Le snapshot de volume est pris en charge par `ontap-nas`, `ontap-nas-flexgroup`, `ontap-san`, `ontap-san-economy`, `solidfire-san`, `gcp-cvs`, et `azure-netapp-files` pilotes.

Avant de commencer

Vous devez disposer d'un contrôleur de snapshot externe et de définitions de ressources personnalisées (CRD). Cela relève de la responsabilité de l'orchestrateur Kubernetes (par exemple : Kubeadm, GKE, OpenShift).

Si votre distribution Kubernetes n'inclut pas le contrôleur de snapshot et les CRD, reportez-vous à la Déploiement d'un contrôleur de snapshot de volume.

> Ne créez pas de contrôleur de snapshot si vous créez des snapshots de volume à la demande dans un environnement GKE. GKE utilise un contrôleur de snapshot caché intégré.

Étape 1 : créer un `VolumeSnapshotClass`

Cet exemple crée une classe de snapshot de volume.

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

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

Pour plus d'informations, consultez le lien :../trident-Reference/objects.html#kubernetes-volumesnapshotclass-objects[`VolumeSnapshotClass`].

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

Dans cet exemple, nous allons créer un snapshot d'un volume persistant existant.

```
cat snap.yaml
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshot
metadata:
  name: pvc1-snap
spec:
  volumeSnapshotClassName: csi-snapclass
  source:
    persistentVolumeClaimName: pvc1
```

Dans cet exemple, le snapshot est créé pour une demande de volume persistant nommée `pvc1` et le nom du snapshot est défini sur `pvc1-snap`.

```
kubectl create -f snap.yaml
volumesnapshot.snapshot.storage.k8s.io/pvc1-snap created

kubectl get volumesnapshots
NAME                   AGE
pvc1-snap              50s
```

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

Il est possible d'identifier le `VolumeSnapshotContent` objet pour le `pvc1-snap` VolumeSnapshot en le décrivant.

```
kubectl describe volumesnapshots pvc1-snap
Name:         pvc1-snap
Namespace:    default
.
.
.
Spec:
  Snapshot Class Name:    pvc1-snap
  Snapshot Content Name:  snapcontent-e8d8a0ca-9826-11e9-9807-525400f3f660
  Source:
    API Group:
    Kind:       PersistentVolumeClaim
    Name:       pvc1
Status:
  Creation Time:  2019-06-26T15:27:29Z
  Ready To Use:   true
  Restore Size:   3Gi
.
.
```

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

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

Dans cet exemple, vous créez une demande de volume persistant à l'aide d'un snapshot :

```
cat pvc-from-snap.yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-from-snap
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: golden
  resources:
    requests:
      storage: 3Gi
  dataSource:
    name: pvc1-snap
    kind: VolumeSnapshot
    apiGroup: snapshot.storage.k8s.io
```

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

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

Déploiement d'un contrôleur de snapshot de volume

Si votre distribution Kubernetes n'inclut pas le contrôleur de snapshot et les CRD, vous pouvez les déployer comme suit.

Étapes

1. Création de CRD de snapshot de volume.

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

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

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-6.1/deploy/kubernetes/snapshot-
controller/rbac-snapshot-controller.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-6.1/deploy/kubernetes/snapshot-
controller/setup-snapshot-controller.yaml
```

Liens connexes

- "Snapshots de volume"
- "VolumeSnapshotClass"

Développement des volumes

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

Développez un volume iSCSI

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

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

Présentation

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

- Modification de la définition de classe de stockage pour définir le `allowVolumeExpansion` champ à

`true`.

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

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

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

```
cat storageclass-ontapsan.yaml
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-san
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
allowVolumeExpansion: True
```

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

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

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

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

```
kubectl get pvc
NAME      STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS   AGE
san-pvc   Bound    pvc-8a814d62-bd58-4253-b0d1-82f2885db671   1Gi
RWO            ontap-san      8s

kubectl get pv
NAME                                       CAPACITY   ACCESS MODES
RECLAIM POLICY   STATUS   CLAIM             STORAGECLASS   REASON   AGE
pvc-8a814d62-bd58-4253-b0d1-82f2885db671   1Gi        RWO
Delete           Bound    default/san-pvc   ontap-san               10s
```

Étape 3 : définissez un pod qui fixe la demande de volume persistant

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

```
 kubectl get pod
NAME         READY   STATUS    RESTARTS   AGE
ubuntu-pod   1/1     Running   0          65s

 kubectl describe pvc san-pvc
Name:          san-pvc
Namespace:     default
StorageClass:  ontap-san
Status:        Bound
Volume:        pvc-8a814d62-bd58-4253-b0d1-82f2885db671
Labels:        <none>
Annotations:   pv.kubernetes.io/bind-completed: yes
               pv.kubernetes.io/bound-by-controller: yes
               volume.beta.kubernetes.io/storage-provisioner:
csi.trident.netapp.io
Finalizers:    [kubernetes.io/pvc-protection]
Capacity:      1Gi
Access Modes:  RWO
VolumeMode:    Filesystem
Mounted By:    ubuntu-pod
```

Étape 4 : développez le volume persistant

Pour redimensionner la PV créée de 1Gi à 2Gi, modifiez la définition de la demande de volume persistant et mettez à jour la `spec.resources.requests.storage` À 2Gi.

```
kubectl edit pvc san-pvc
# Please edit the object below. Lines beginning with a '#' will be
ignored,
# and an empty file will abort the edit. If an error occurs while saving
this file will be
# reopened with the relevant failures.
#
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    pv.kubernetes.io/bind-completed: "yes"
    pv.kubernetes.io/bound-by-controller: "yes"
    volume.beta.kubernetes.io/storage-provisioner: csi.trident.netapp.io
  creationTimestamp: "2019-10-10T17:32:29Z"
  finalizers:
  - kubernetes.io/pvc-protection
  name: san-pvc
  namespace: default
  resourceVersion: "16609"
  selfLink: /api/v1/namespaces/default/persistentvolumeclaims/san-pvc
  uid: 8a814d62-bd58-4253-b0d1-82f2885db671
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 2Gi
 ...
```

Étape 5 : valider l'extension

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

```
kubectl get pvc san-pvc
NAME      STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS   AGE
san-pvc   Bound    pvc-8a814d62-bd58-4253-b0d1-82f2885db671   2Gi
RWO            ontap-san      11m
kubectl get pv
NAME                                       CAPACITY   ACCESS MODES
RECLAIM POLICY   STATUS   CLAIM             STORAGECLASS   REASON   AGE
pvc-8a814d62-bd58-4253-b0d1-82f2885db671   2Gi        RWO
Delete           Bound    default/san-pvc   ontap-san               12m
tridentctl get volumes -n trident
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
|                   NAME                  |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
| pvc-8a814d62-bd58-4253-b0d1-82f2885db671 | 2.0 GiB | ontap-san     |
block    | a9b7bfff-0505-4e31-b6c5-59f492e02d33 | online | true    |
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
```

Développez un volume NFS

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

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

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

```
cat storageclass-ontapnas.yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontapnas
provisioner: csi.trident.netapp.io
parameters:
  backendType: ontap-nas
allowVolumeExpansion: true
```

Si vous avez déjà créé une classe de stockage sans cette option, vous pouvez simplement modifier la classe de stockage existante en utilisant `kubectl edit storageclass` pour permettre l'extension de volume.

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

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

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

```
kubectl get pvc
NAME           STATUS   VOLUME
CAPACITY     ACCESS MODES   STORAGECLASS    AGE
ontapnas20mb   Bound    pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7   20Mi
RWO            ontapnas        9s

kubectl get pv pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7
NAME                                       CAPACITY   ACCESS MODES
RECLAIM POLICY   STATUS   CLAIM                  STORAGECLASS    REASON
AGE
pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7   20Mi       RWO
Delete           Bound    default/ontapnas20mb   ontapnas
2m42s
```

Étape 3 : développez le volume persistant

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

```
kubectl edit pvc ontapnas20mb
# Please edit the object below. Lines beginning with a '#' will be
ignored,
# and an empty file will abort the edit. If an error occurs while saving
this file will be
# reopened with the relevant failures.
#
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    pv.kubernetes.io/bind-completed: "yes"
    pv.kubernetes.io/bound-by-controller: "yes"
    volume.beta.kubernetes.io/storage-provisioner: csi.trident.netapp.io
  creationTimestamp: 2018-08-21T18:26:44Z
  finalizers:
  - kubernetes.io/pvc-protection
  name: ontapnas20mb
  namespace: default
  resourceVersion: "1958015"
  selfLink: /api/v1/namespaces/default/persistentvolumeclaims/ontapnas20mb
  uid: c1bd7fa5-a56f-11e8-b8d7-fa163e59eaab
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
...
```

Étape 4 : valider l'extension

Vous pouvez valider le redimensionnement correctement en contrôlant la taille de la demande de volume persistant, de la volume persistant et du volume Astra Trident :

```
kubectl get pvc ontapnas20mb
NAME           STATUS   VOLUME
CAPACITY   ACCESS MODES   STORAGECLASS    AGE
ontapnas20mb   Bound    pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7   1Gi
RWO            ontapnas        4m44s

kubectl get pv pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7
NAME                                       CAPACITY   ACCESS MODES
RECLAIM POLICY   STATUS   CLAIM                  STORAGECLASS    REASON
AGE
pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7   1Gi        RWO
Delete           Bound    default/ontapnas20mb   ontapnas
5m35s

tridentctl get volume pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7 -n trident
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
|                   NAME                  |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID              | STATE  | MANAGED |
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
| pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7 | 1.0 GiB | ontapnas      |
file     | c5a6f6a4-b052-423b-80d4-8fb491a14a22 | online | true    |
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
```

Importer des volumes

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

Pilotes prenant en charge l'importation de volumes

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

Conducteur	Relâchez
`ontap-nas`	19.04
`ontap-nas-flexgroup`	19.04
`solidfire-san`	19.04
`azure-netapp-files`	19.04

Conducteur	Relâchez
gcp-cvs	19.04
ontap-san	19.04

Pourquoi importer des volumes ?

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

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

Comment fonctionne l'importation ?

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

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

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

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

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

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

le volume persistant, la règle de récupération est mise à jour pour correspondre à la règle de récupération de la classe de stockage. Si la règle de récupération de la classe de stockage est `delete`, Le volume de stockage sera supprimé lorsque le volume persistant est supprimé.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

`ontap-san` importer

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

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

- Indiquez le nom de la FlexVol à importer. N'oubliez pas que cette FlexVol ne contient qu'une seule LUN qui doit être importée.
- Fournir le chemin de la définition de PVC qui doit être utilisée avec le `-f` drapeau.
- Vous avez le choix entre gérer ou non le volume persistant. Par défaut, Trident gère le volume de volume persistant et renomme la FlexVol et la LUN en back-end. Pour importer en tant que volume non géré, passez le `--no-manage` drapeau.

Lors de l'importation d'un non géré `ontap-san` Volume, vérifiez que la LUN de la FlexVol est nommée `lun0` et est mappée sur un groupe initiateur avec les initiateurs souhaités. Astra Trident le gère automatiquement pour une importation gérée.

Astra Trident va ensuite importer le FlexVol et l'associer à la définition de la demande de volume persistant. Astra Trident renomme également le FlexVol avec le `pvc-<uuid>` Formatez et la LUN au sein du FlexVol à `lun0`.

Il est recommandé d'importer des volumes qui n'ont pas de connexions actives existantes. Pour importer un volume activement utilisé, commencez par cloner le volume, puis procédez à l'importation.

Exemple

Pour importer `ontap-san-managed` FlexVol présent sur le `ontap_san_default` back-end, exécutez le `tridentctl import` sous forme de commande :

```
tridentctl import volume ontapsan_san_default ontap-san-managed -f pvc-
basic-import.yaml -n trident -d

+-----------------------------------------+--------+--------------
+----------+-----------------------------------------+--------+---------+
|                   NAME                  |  SIZE  | STORAGE CLASS |
PROTOCOL |             BACKEND UUID              | STATE  | MANAGED |
+-----------------------------------------+--------+--------------
+----------+-----------------------------------------+--------+---------+
| pvc-d6ee4f54-4e40-4454-92fd-d00fc228d74a | 20 MiB | basic         |
block    | cd394786-ddd5-4470-adc3-10c5ce4ca757 | online | true    |
+-----------------------------------------+--------+--------------
+----------+-----------------------------------------+--------+---------+
```

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

`element` importer

Vous pouvez importer le logiciel NetApp Element/les volumes NetApp HCI dans votre cluster Kubernetes avec Trident. Vous avez besoin du nom de votre back-end Astra Trident, ainsi que du nom unique du volume et du fichier PVC comme arguments pour le `tridentctl import` commande.

```
tridentctl import volume element_default element-managed -f pvc-basic-
import.yaml -n trident -d

+-----------------------------------------+--------+--------------
+----------+-----------------------------------------+--------+---------+
|                   NAME                  |  SIZE  | STORAGE CLASS |
PROTOCOL |             BACKEND UUID              | STATE  | MANAGED |
+-----------------------------------------+--------+--------------
+----------+-----------------------------------------+--------+---------+
| pvc-970ce1ca-2096-4ecd-8545-ac7edc24a8fe | 10 GiB | basic-element |
block    | d3ba047a-ea0b-43f9-9c42-e38e58301c49 | online | true    |
+-----------------------------------------+--------+--------------
+----------+-----------------------------------------+--------+---------+
```

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

`gcp-cvs` importer

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

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

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

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

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

`azure-netapp-files` **importer**

Pour importer un `azure-netapp-files` volume sur le back-end appelé `azurenetappfiles_40517` avec le chemin de volume `importvol1`, exécutez la commande suivante :

```
tridentctl import volume azurenetappfiles_40517 importvol1 -f <path-to-
pvc-file> -n trident

+----------------------------------------+---------+---------------
+----------+----------------------------------------+---------+---------+
|                   NAME                 |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID             | STATE   | MANAGED |
+----------------------------------------+---------+---------------
+----------+----------------------------------------+---------+---------+
| pvc-0ee95d60-fd5c-448d-b505-b72901b3a4ab | 100 GiB | anf-storage   |
file     | 1c01274f-d94b-44a3-98a3-04c953c9a51e | online | true    |
+----------------------------------------+---------+---------------
+----------+----------------------------------------+---------+---------+
```

Le chemin de volume du volume ANF est présent dans le chemin de montage après :/. Par exemple, si le chemin de montage est `10.0.0.2:/importvol1`, le chemin du volume est `importvol1`.

Partager un volume NFS entre les espaces de noms

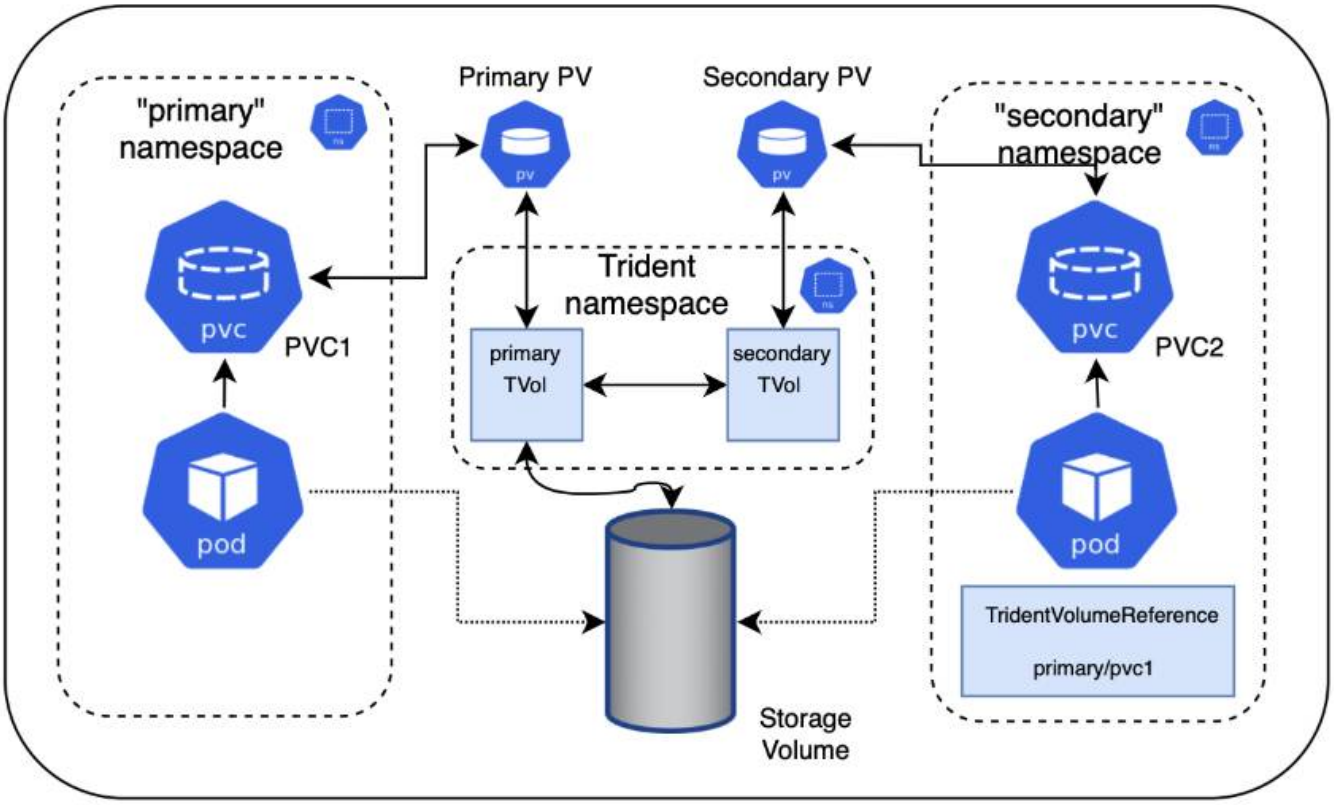
Avec Astra Trident, vous pouvez créer un volume dans un espace de noms principal et le partager dans un ou plusieurs espaces de noms secondaires.

Caractéristiques

Le système Astra TridentVolumeReference CR vous permet de partager en toute sécurité des volumes NFS ReadWriteMany (RWX) sur un ou plusieurs espaces de noms Kubernetes. Cette solution Kubernetes-native présente plusieurs avantages :

- Plusieurs niveaux de contrôle d'accès pour assurer la sécurité
- Fonctionne avec tous les pilotes de volume NFS Trident
- Pas de dépendance à tridentctl ou à toute autre fonctionnalité Kubernetes non native

Ce schéma illustre le partage de volumes NFS entre deux espaces de noms Kubernetes.



Démarrage rapide

Vous pouvez configurer le partage de volumes NFS en quelques étapes seulement.

1 **Configurez la demande de volume persistant source pour partager le volume**

Le propriétaire de l'espace de noms source autorise l'accès aux données dans la demande de volume persistant source.

2 **Accorder l'autorisation de créer une demande de modification dans l'espace de noms de destination**

L'administrateur de cluster accorde l'autorisation au propriétaire de l'espace de noms de destination pour créer le CR TridentVolumeReference.

3 **Créer TridentVolumeReference dans l'espace de noms de destination**

Le propriétaire de l'espace de noms de destination crée le CR TridentVolumeReference pour faire référence au PVC source.

4 **Créez le PVC subalterne dans l'espace de noms de destination**

Le propriétaire de l'espace de noms de destination crée le PVC subalterne pour utiliser la source de données à partir du PVC source.

Configurer les espaces de noms source et de destination

Pour garantir la sécurité, le partage de l'espace de noms croisé nécessite une collaboration et une action du propriétaire de l'espace de noms source, de l'administrateur de cluster et du propriétaire de l'espace de noms de destination. Le rôle utilisateur est désigné dans chaque étape.

Étapes

1. **Propriétaire de l'espace de noms source:** Créez le PVC (`pvc1`) dans l'espace de noms source qui accorde l'autorisation de partager avec l'espace de noms de destination (`namespace2`) à l'aide de l' `shareToNamespace` annotation.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc1
  namespace: namespace1
  annotations:
    trident.netapp.io/shareToNamespace: namespace2
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: trident-csi
  resources:
    requests:
      storage: 100Gi
```

   Astra Trident crée le volume persistant et son volume de stockage NFS back-end.

   - Vous pouvez partager le PVC sur plusieurs espaces de noms à l'aide d'une liste délimitée par des virgules. Par exemple : `trident.netapp.io/shareToNamespace: namespace2,namespace3,namespace4`.
   - Vous pouvez partager avec tous les espaces de noms à l'aide de `*`. Par exemple : `trident.netapp.io/shareToNamespace: *`
   - Vous pouvez mettre à jour le PVC pour inclure le `shareToNamespace` annotation à tout moment.

2. **Cluster admin:** Créez le rôle personnalisé et kubeconfig pour accorder l'autorisation au propriétaire de l'espace de noms de destination de créer le CR TridentVolumeReference dans l'espace de noms de destination.
3. **Propriétaire de l'espace de noms de destination :** Créez un CR TridentVolumeReference dans l'espace de noms de destination qui fait référence à l'espace de noms source `pvc1`.

```
apiVersion: trident.netapp.io/v1
kind: TridentVolumeReference
metadata:
  name: my-first-tvr
  namespace: namespace2
spec:
  pvcName: pvc1
  pvcNamespace: namespace1
```

4. **Propriétaire de l'espace de noms de destination:** Créer un PVC (`pvc2`) dans l'espace de noms de destination (`namespace2`) à l'aide de l' `shareFromPVC` Annotation pour désigner la PVC source.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  annotations:
    trident.netapp.io/shareFromPVC: namespace1/pvc1
  name: pvc2
  namespace: namespace2
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: trident-csi
  resources:
    requests:
      storage: 100Gi
```

La taille du PVC de destination doit être inférieure ou égale à la PVC source.

Résultats

Découvrez Astra Trident `shareFromPVC` Annotation sur la demande de volume persistant de destination et création du volume persistant de destination en tant que volume subalterne, sans ressource de stockage correspondant au volume persistant source et partage la ressource de stockage PV source. La demande de volume persistant et la demande de volume persistant de destination apparaissent comme normales.

Supprimer un volume partagé

Vous pouvez supprimer un volume partagé entre plusieurs namespaces. Astra Trident supprimera l'accès au volume de l'espace de noms source et maintiendra l'accès aux autres espaces de noms qui partagent le volume. Lorsque tous les namespaces qui référencent le volume sont supprimés, Astra Trident supprime le volume.

Utiliser `tridentctl get` pour interroger des volumes subordonnés

À l'aide du[`tridentctl` vous pouvez exécuter l' `get` commande pour obtenir des volumes subordonnés.

Pour plus d'informations, consultez le lien :../trident-Reference/tridentctl.html[`tridentctl` commandes et options].

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

Alarmes :

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

Limites

- Astra Trident ne peut pas empêcher les espaces de noms de destination d'écrire sur le volume partagé. Nous vous recommandons d'utiliser un verrouillage de fichiers ou d'autres processus pour éviter d'écraser les données du volume partagé.
- Vous ne pouvez pas révoquer l'accès au PVC source en retirant le `shareToNamespace` ou `shareFromNamespace` annotations ou suppression du `TridentVolumeReference` CR. Pour annuler l'accès, vous devez supprimer le PVC subalterne.
- Les snapshots, clones et la mise en miroir ne sont pas possibles sur les volumes subordonnés.

Pour en savoir plus

Pour en savoir plus sur l'accès aux volumes multi-espaces de noms :

- Visitez "Partage de volumes entre les espaces de noms : dites bonjour à l'accès aux volumes situés à l'échelle d'un espace de noms".
- Regardez la démo sur "NetAppTV".

Contrôle d'Astra Trident

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

Avec les metrics d'Astra Trident, vous pouvez :

- Surveillez l'état et la configuration d'Astra Trident. Vous avez la possibilité d'examiner la réussite des opérations et de savoir si elles peuvent communiquer avec les systèmes back-end comme prévu.
- Examiner les informations d'utilisation du système back-end et comprendre le nombre de volumes provisionnés sur un système back-end, ainsi que la quantité d'espace consommé, etc.
- Conservez un mappage de la quantité de volumes provisionnés sur les systèmes back-end disponibles.
- Suivi des performances. Découvrez le temps nécessaire pour communiquer avec Astra Trident aux systèmes back-end et effectuer les opérations.

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

Ce dont vous avez besoin

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

Étape 1 : définir une cible Prometheus

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

Étape 2 : créer un ServiceMonitor Prometheus

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

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

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

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

Étape 3 : interroger les mesures Trident avec PromQL

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

Voici quelques questions PromQL que vous pouvez utiliser :

Accédez aux informations sur l'état de santé de Trident

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

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

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

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

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

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

Découvrez les informations d'utilisation d'Astra Trident

- **Taille moyenne du volume**

```
trident_volume_allocated_bytes/trident_volume_count
```

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

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

Utiliser individuellement le volume

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

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

```
kubelet_volume_stats_used_bytes / kubelet_volume_stats_capacity_bytes *
100
```

Découvrez la télémétrie AutoSupport d'Astra Trident

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

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

Voici un exemple de charge utile envoyée par Astra Trident :

```
---
items:
- backendUUID: ff3852e1-18a5-4df4-b2d3-f59f829627ed
  protocol: file
  config:
    version: 1
    storageDriverName: ontap-nas
    debug: false
    debugTraceFlags:
    disableDelete: false
    serialNumbers:
    - nwkvzfanek_SN
    limitVolumeSize: ''
  state: online
  online: true
```

- Les messages AutoSupport sont envoyés au terminal AutoSupport de NetApp. Si vous utilisez un registre privé pour stocker des images de conteneur, vous pouvez utiliser le `--image-registry` drapeau.
- Vous pouvez également configurer des URL proxy en générant les fichiers YAML d'installation. Pour ce

faire, utilisez `tridentctl install --generate-custom-yaml` Pour créer les fichiers YAML et ajouter le `--proxy-url` argument pour le `trident-autosupport` conteneur `trident-deployment.yaml`.

Désactivation des metrics d'Astra Trident

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

Astra Trident pour Docker

Conditions préalables au déploiement

Vous devez installer et configurer les protocoles requis sur votre hôte avant de déployer Astra Trident.

Vérifier les exigences

- Vérifiez que votre déploiement répond à toutes les "de formation".
- Vérifiez que vous disposez d'une version prise en charge de Docker installée. Si votre version de Docker est obsolète, "installez-le ou mettez-le à jour".

```
docker --version
```

- Vérifiez que les conditions préalables au protocole sont installées et configurées sur votre hôte :

Protocole	Système d'exploitation	Commandes
NFS	RHEL 8+	`sudo yum install -y nfs-utils`
NFS	Ubuntu	`sudo apt-get install -y nfs-common`

Protocole	Système d'exploitation	Commandes

ISCSI	RHEL 8+	1. Installez les packages système suivants :
		
		`sudo yum install -y lsscsi iscsi-initiator-utils sg3_utils device-mapper-multipath`
		
		2. Vérifiez que la version iscsi-initiator-utils est 6.2.0.874-2.el7 ou ultérieure :
		
		`rpm -q iscsi-initiator-utils`
		
		3. Définir la numérisation sur manuelle :
		
		`sudo sed -i 's/^\(node.session.scan\).*/\1 = manual/' /etc/iscsi/iscsid.conf`
		
		4. Activer les chemins d'accès multiples :
		
		`sudo mpathconf --enable --with_multipathd y --find_multipaths n`
		
		Bien sûr `etc/multipath.conf` contient `find_multipaths no` sous `defaults`.
		
		5. S'assurer que `iscsid` et `multipathd` sont en cours d'exécution :
		
		`sudo systemctl enable --now iscsid multipathd`
		
		6. Activer et démarrer `iscsi`:
		
		`sudo systemctl enable --now iscsi`

Protocole	Système d'exploitation	Commandes

ISCSI	Ubuntu	1. Installez les packages système suivants :
		
		`sudo apt-get install -y open-iscsi lsscsi sg3-utils multipath-tools scsitools`
		
		2. Vérifiez que la version Open-iscsi est 2.0.874-5ubuntu2.10 ou ultérieure (pour bionique) ou 2.0.874-7.1ubuntu6.1 ou ultérieure (pour focaux) :
		
		`dpkg -l open-iscsi`
		
		3. Définir la numérisation sur manuelle :
		
		`sudo sed -i 's/^\(node.session.scan\).*/\1 = manual/' /etc/iscsi/iscsid.conf`
		
		4. Activer les chemins d'accès multiples :
		
		`sudo tee /etc/multipath.conf <←'EOF'`
		`defaults {`
		`user_friendly_names yes`
		`find_multipaths no`
		`}`
		`EOF`
		`sudo systemctl enable --now multipath-tools.service`
		`sudo service multipath-tools restart`
		
		Bien sûr `etc/multipath.conf` contient `find_multipaths no` sous `defaults`.
		
		5. S'assurer que `open-iscsi` et `multipath-tools` sont activées et en cours d'exécution :
		
		`sudo systemctl status multipath-tools`

Déployez Astra Trident

Astra Trident pour Docker offre une intégration directe avec l'écosystème Docker pour les plateformes de stockage NetApp. Il prend en charge le provisionnement et la gestion des ressources de stockage, depuis la plateforme de stockage jusqu'aux hôtes Docker, par exemple, l'ajout de plateformes supplémentaires à l'avenir.

Plusieurs instances d'Astra Trident peuvent être exécutées simultanément sur le même hôte. Vous pouvez ainsi établir des connexions simultanées à plusieurs systèmes et types de stockage, et personnaliser le stockage utilisé pour les volumes Docker.

Ce dont vous avez besoin

Voir la "conditions préalables au déploiement". Une fois que vous avez rempli les conditions préalables, vous êtes prêt à déployer Astra Trident.

Méthode de plug-in géré Docker (version 1.13/17.03 et ultérieure)

Avant de commencer

Si vous avez utilisé Astra Trident pré Docker 1.13/17.03 dans la méthode du démon traditionnel, veillez à arrêter le processus Astra Trident et à redémarrer votre démon Docker avant d'utiliser la méthode du plug-in géré.

1. Arrêter toutes les instances en cours d'exécution :

```
pkill /usr/local/bin/netappdvp
pkill /usr/local/bin/trident
```

2. Redémarrez Docker.

```
systemctl restart docker
```

3. Vérifiez que Docker Engine 17.03 (nouveau modèle 1.13) ou ultérieur est installé.

```
docker --version
```

Si votre version est obsolète, "installez ou mettez à jour votre installation".

Étapes

1. Créez un fichier de configuration et spécifiez les options comme suit :
   - `config`: Le nom de fichier par défaut est `config.json`, cependant, vous pouvez utiliser un nom quelconque en spécifiant le `config` avec le nom de fichier. Le fichier de configuration doit se trouver dans le `/etc/netappdvp` répertoire sur le système hôte.
   - `log-level`: Spécifiez le niveau de consignation (`debug`, `info`, `warn`, `error`, `fatal`). La valeur par défaut est `info`.

◦ `debug`: Spécifiez si la journalisation de débogage est activée. La valeur par défaut est FALSE. Remplace le niveau de journalisation si vrai.

i. Créez un emplacement pour le fichier de configuration :

```
sudo mkdir -p /etc/netappdvp
```

ii. Créez le fichier de configuration :

```
cat << EOF > /etc/netappdvp/config.json
{
    "version": 1,
    "storageDriverName": "ontap-nas",
    "managementLIF": "10.0.0.1",
    "dataLIF": "10.0.0.2",
    "svm": "svm_nfs",
    "username": "vsadmin",
    "password": "password",
    "aggregate": "aggr1"
}
EOF
```

2. Démarrez Astra Trident à l'aide du système de plug-in géré. Remplacement `<version>` avec la version du plugin (xxx.xx.x) que vous utilisez.

```
docker plugin install --grant-all-permissions --alias netapp
netapp/trident-plugin:<version> config=myConfigFile.json
```

3. Commencez à utiliser Astra Trident pour consommer le stockage à partir du système configuré.

a. Créer un volume nommé « firstVolume » :

```
docker volume create -d netapp --name firstVolume
```

b. Créez un volume par défaut au démarrage du conteneur :

```
docker run --rm -it --volume-driver netapp --volume
secondVolume:/my_vol alpine ash
```

c. Supprimez le volume « firstVolume » :

```
docker volume rm firstVolume
```

Méthode traditionnelle (version 1.12 ou antérieure)

Avant de commencer

1. Vérifiez que Docker version 1.10 ou ultérieure est installé.

   ```
   docker --version
   ```

   Si votre version est obsolète, mettez à jour votre installation.

   ```
   curl -fsSL https://get.docker.com/ | sh
   ```

   Ou "suivez les instructions relatives à votre distribution".

2. Vérifiez que NFS et/ou iSCSI sont configurés pour votre système.

Étapes

1. Installez et configurez le plug-in de volume NetApp Docker :
   a. Téléchargez et déballez l'application :

      ```
      wget
      https://github.com/NetApp/trident/releases/download/v23.01.1/trident-
      installer-23.01.1.tar.gz
      tar zxf trident-installer-23.01.1.tar.gz
      ```

   b. Déplacer vers un emplacement dans le chemin du bac :

      ```
      sudo mv trident-installer/extras/bin/trident /usr/local/bin/
      sudo chown root:root /usr/local/bin/trident
      sudo chmod 755 /usr/local/bin/trident
      ```

   c. Créez un emplacement pour le fichier de configuration :

      ```
      sudo mkdir -p /etc/netappdvp
      ```

   d. Créez le fichier de configuration :

```
cat << EOF > /etc/netappdvp/ontap-nas.json
{
    "version": 1,
    "storageDriverName": "ontap-nas",
    "managementLIF": "10.0.0.1",
    "dataLIF": "10.0.0.2",
    "svm": "svm_nfs",
    "username": "vsadmin",
    "password": "password",
    "aggregate": "aggr1"
}
EOF
```

2. Après avoir placé le binaire et créé le(s) fichier(s) de configuration, démarrez le démon Trident à l'aide du fichier de configuration souhaité.

```
sudo trident --config=/etc/netappdvp/ontap-nas.json
```

Sauf indication contraire, le nom par défaut du pilote de volume est « netapp ».

Une fois le démon démarré, vous pouvez créer et gérer des volumes à l'aide de l'interface de ligne de commande de Docker

3. Créer un volume :

```
docker volume create -d netapp --name trident_1
```

4. Provisionnement d'un volume Docker lors du démarrage d'un conteneur :

```
docker run --rm -it --volume-driver netapp --volume trident_2:/my_vol
alpine ash
```

5. Supprimer un volume Docker :

```
docker volume rm trident_1
docker volume rm trident_2
```

Commencez avec Astra Trident au démarrage du système

Un exemple de fichier d'unité pour les systèmes basés sur le système se trouve à l'adresse `contrib/trident.service.example` Dans le Git repo. Pour utiliser le fichier avec RHEL, procédez comme suit :

1. Copiez le fichier à l'emplacement correct.

   Vous devez utiliser des noms uniques pour les fichiers d'unité si plusieurs instances sont en cours d'exécution.

   ```
   cp contrib/trident.service.example
   /usr/lib/systemd/system/trident.service
   ```

2. Modifiez le fichier, modifiez la description (ligne 2) pour qu'elle corresponde au nom du pilote et au chemin du fichier de configuration (ligne 9) pour qu'elle corresponde à votre environnement.
3. Recharger le système pour qu'il ingère les modifications :

   ```
   systemctl daemon-reload
   ```

4. Activer le service.

   Ce nom varie en fonction de ce que vous avez nommé le fichier dans le `/usr/lib/systemd/system` répertoire.

   ```
   systemctl enable trident
   ```

5. Démarrer le service.

   ```
   systemctl start trident
   ```

6. Afficher l'état.

   ```
   systemctl status trident
   ```

Chaque fois que vous modifiez le fichier d'unité, exécutez le `systemctl daemon-reload` commande pour que le service it soit conscient des modifications.

Mise à niveau ou désinstallation d'Astra Trident

Vous pouvez mettre à niveau Astra Trident pour Docker en toute sécurité, sans impact sur les volumes en cours d'utilisation. Pendant le processus de mise à niveau, il y aura une courte période où `docker volume` les commandes dirigées au niveau du plug-in ne réussiront pas et les applications ne pourront pas monter les volumes tant que le plug-in ne sera pas de nouveau exécuté. Dans la plupart des cas, c'est une question de secondes.

Mise à niveau

Suivez les étapes ci-dessous pour mettre à niveau Astra Trident pour Docker.

Étapes

1. Lister les volumes existants :

```
docker volume ls
DRIVER              VOLUME NAME
netapp:latest       my_volume
```

2. Désactivez le plug-in :

```
docker plugin disable -f netapp:latest
docker plugin ls
ID                  NAME                DESCRIPTION
ENABLED
7067f39a5df5        netapp:latest       nDVP - NetApp Docker Volume
Plugin   false
```

3. Mettez à niveau le plug-in :

```
docker plugin upgrade --skip-remote-check --grant-all-permissions
netapp:latest netapp/trident-plugin:21.07
```

> La version 18.01 d'Astra Trident remplace le nDVP. Vous devez mettre à niveau directement à partir du `netapp/ndvp-plugin` image vers le `netapp/trident-plugin` image.

4. Activer le plug-in :

```
docker plugin enable netapp:latest
```

5. Vérifiez que le plug-in est activé :

```
docker plugin ls
ID                  NAME                DESCRIPTION
ENABLED
7067f39a5df5        netapp:latest       Trident - NetApp Docker Volume
Plugin   true
```

6. Vérifier que les volumes sont visibles :

```
docker volume ls
DRIVER              VOLUME NAME
netapp:latest       my_volume
```

Si vous effectuez la mise à niveau d'une ancienne version d'Astra Trident (pré-20.10) vers Astra Trident 20.10 ou version ultérieure, vous risquez de vous produire une erreur. Pour plus d'informations, voir "Problèmes connus". Si vous exécutez l'erreur, vous devez d'abord désactiver le plug-in, puis retirer le plug-in, puis installer la version Astra Trident requise en passant un paramètre de configuration supplémentaire : `docker plugin install netapp/trident-plugin:20.10 --alias netapp --grant-all-permissions config=config.json`

Désinstaller

Effectuez les opérations suivantes pour désinstaller Astra Trident pour Docker.

Étapes

1. Supprimez tous les volumes créés par le plug-in.
2. Désactivez le plug-in :

```
docker plugin disable netapp:latest
docker plugin ls
ID                  NAME                DESCRIPTION
ENABLED
7067f39a5df5        netapp:latest       nDVP - NetApp Docker Volume
Plugin   false
```

3. Retirez le plug-in :

```
docker plugin rm netapp:latest
```

Utilisation de volumes

Vous pouvez facilement créer, cloner et supprimer des volumes à l'aide de la norme `docker volume` Commandes avec le nom de pilote Astra Trident spécifié le cas échéant.

Créer un volume

- Créez un volume avec un pilote à l'aide du nom par défaut :

```
docker volume create -d netapp --name firstVolume
```

- Créez un volume avec une instance Astra Trident spécifique :

```
docker volume create -d ntap_bronze --name bronzeVolume
```

Si vous n'en spécifiez aucun "options", les valeurs par défaut du pilote sont utilisées.

- Remplacer la taille du volume par défaut. Voir l'exemple suivant pour créer un volume de 20 Gio avec un pilote :

```
docker volume create -d netapp --name my_vol --opt size=20G
```

Les tailles de volume sont exprimées en chaînes contenant une valeur entière avec des unités facultatives (par exemple : 10G, 20 Go, Tio). Si aucune unité n'est spécifiée, la valeur par défaut est G. Les unités de taille peuvent être exprimées en puissances de 2 (B, Kio, Mio, Gio, Tio) ou 10 (B, Ko, Mo, Go, To). Les unités de raccourci utilisent des puissances de 2 (G = Gio, T = Tio, …).

Supprimer un volume

- Supprimez le volume comme n'importe quel autre volume Docker :

```
docker volume rm firstVolume
```

Lorsque vous utilisez le `solidfire-san` pilote, l'exemple ci-dessus supprime et purge le volume.

Suivez les étapes ci-dessous pour mettre à niveau Astra Trident pour Docker.

Clonez un volume

Lorsque vous utilisez le `ontap-nas`, `ontap-san`, `solidfire-san`, et `gcp-cvs storage drivers`, Astra Trident peut cloner des volumes. Lorsque vous utilisez le `ontap-nas-flexgroup` ou `ontap-nas-economy` le clonage des pilotes n'est pas pris en charge. La création d'un nouveau volume à partir d'un volume existant entraîne la création d'un nouveau snapshot.

- Inspectez le volume pour énumérer les instantanés :

```
docker volume inspect <volume_name>
```

- Créer un nouveau volume à partir d'un volume existant. Cela entraîne la création d'un nouvel instantané :

```
docker volume create -d <driver_name> --name <new_name> -o
from=<source_docker_volume>
```

- Créer un nouveau volume à partir d'un snapshot existant sur un volume. Cette opération ne crée pas de nouvel instantané :

```
docker volume create -d <driver_name> --name <new_name> -o
from=<source_docker_volume> -o fromSnapshot=<source_snap_name>
```

Exemple

```
docker volume inspect firstVolume

[
    {
        "Driver": "ontap-nas",
        "Labels": null,
        "Mountpoint": "/var/lib/docker-volumes/ontap-
nas/netappdvp_firstVolume",
        "Name": "firstVolume",
        "Options": {},
        "Scope": "global",
        "Status": {
            "Snapshots": [
                {
                    "Created": "2017-02-10T19:05:00Z",
                    "Name": "hourly.2017-02-10_1505"
                }
            ]
        }
    }
]

docker volume create -d ontap-nas --name clonedVolume -o from=firstVolume
clonedVolume

docker volume rm clonedVolume
docker volume create -d ontap-nas --name volFromSnap -o from=firstVolume
-o fromSnapshot=hourly.2017-02-10_1505
volFromSnap

docker volume rm volFromSnap
```

Accéder aux volumes créés en externe

Vous pouvez accéder aux périphériques de blocs créés en externe (ou à leurs clones) à l'aide de conteneurs utilisant Trident **uniquement**, s'ils ne possèdent aucune partition et si leur système de fichiers est pris en charge par Astra Trident (par exemple, un `ext4`-formaté `/dev/sdc1` Ne sera pas accessible via Astra Trident).

Options de volume spécifiques au conducteur

Chaque pilote de stockage dispose d'un ensemble d'options différent, que vous pouvez spécifier au moment de la création du volume pour personnaliser le résultat. Vous trouverez ci-dessous les options qui s'appliquent à votre système de stockage configuré.

Ces options sont simples à utiliser lors de l'opération de création de volume. Indiquez l'option et la valeur à l'aide de la `-o` Opérateur pendant le fonctionnement de l'interface de ligne de commande. Ces valeurs remplacent toute valeur équivalente du fichier de configuration JSON.

Options de volume ONTAP

Les options de création de volumes pour NFS et iSCSI sont les suivantes :

Option	Description
`size`	La taille du volume est de 1 Gio par défaut.
`spaceReserve`	Provisionnement fin ou non fin du volume, conversion par défaut en fin. Les valeurs valides sont `none` (provisionnement fin) et `volume` (provisionnement lourd).
`snapshotPolicy`	La règle de snapshot sera alors définie sur la valeur souhaitée. La valeur par défaut est `none`, cela signifie qu'aucun instantané ne sera automatiquement créé pour le volume. Sauf modification de la part de votre administrateur de stockage, une règle nommée « par défaut » existe sur tous les systèmes ONTAP qui créent et conserve six snapshots toutes les heures, deux par jour et deux fois par semaine. Vous pouvez restaurer les données conservées dans un snapshot en accédant au `.snapshot` dans n'importe quel répertoire du volume.

Option	Description
`snapshotReserve`	La réserve d'instantanés sera alors définie sur le pourcentage souhaité. La valeur par défaut n'est pas définie. Cela signifie que ONTAP sélectionne la fonction de copie instantanée (généralement 5 %) si vous avez sélectionné une stratégie de snapshots, ou 0 % si la stratégie de snapshots n'est pas définie. Vous pouvez définir la valeur par défaut des snapshots dans le fichier de configuration pour tous les systèmes back-end ONTAP. Vous pouvez l'utiliser comme option de création de volumes pour tous les systèmes back-end ONTAP, à l'exception des économies ontap-nas.
`splitOnClone`	Lors du clonage d'un volume, ONTAP va immédiatement séparer le clone de son volume parent. La valeur par défaut est `false`. Pour optimiser l'efficacité du stockage, il est préférable de séparer le clone de son parent dès sa création, car il est peu probable que cette utilisation soit utile. Par exemple, le clonage d'une base de données vide peut permettre d'économiser beaucoup de temps, mais peu d'espace de stockage économisé, il est donc préférable de séparer immédiatement le clone.
`encryption`	Activez NetApp Volume Encryption (NVE) sur le nouveau volume. La valeur par défaut est `false`. Pour utiliser cette option, NVE doit être sous licence et activé sur le cluster. Si NAE est activé sur le back-end, tous les volumes provisionnés dans Astra Trident seront activés par NAE. Pour plus d'informations, se reporter à : "Fonctionnement d'Astra Trident avec NVE et NAE".
`tieringPolicy`	Définit la règle de hiérarchisation à utiliser pour le volume. Cette décision détermine si les données sont déplacées vers le Tier cloud lorsqu'elles deviennent inactives.

Les options supplémentaires suivantes concernent NFS **uniquement** :

Option	Description
`unixPermissions`	Cette option contrôle les autorisations définies pour le volume lui-même. Par défaut, les autorisations sont définies sur `---rwxr-xr-x`, ou en notation numérique 0755, et `root` sera le propriétaire. Le format texte ou numérique fonctionnera.
`snapshotDir`	Régler sur `true` fera le `.snapshot` répertoire visible par les clients qui accèdent au volume. La valeur par défaut est `false`, ce qui signifie que la visibilité du `.snapshot` le répertoire est désactivé par défaut. Certaines images, par exemple l'image officielle MySQL, ne fonctionnent pas comme prévu lorsque le `.snapshot` le répertoire est visible.
`exportPolicy`	Définit l'export policy à utiliser pour le volume. La valeur par défaut est `default`.
`securityStyle`	Définit le style de sécurité à utiliser pour accéder au volume. La valeur par défaut est `unix`. Les valeurs valides sont `unix` et `mixed`.

Les options supplémentaires suivantes sont disponibles pour iSCSI **uniquement** :

Option	Description
`fileSystemType`	Définit le système de fichiers utilisé pour formater les volumes iSCSI. La valeur par défaut est `ext4`. Les valeurs valides sont `ext3`, `ext4`, et `xfs`.
`spaceAllocation`	Régler sur `false` Désactive la fonction d'allocation d'espace de la LUN. La valeur par défaut est `true`, Qui signifie que ONTAP notifie l'hôte lorsque l'espace du volume est insuffisant et que la LUN du volume ne peut pas accepter les écritures. Cette option permet également à ONTAP de récupérer automatiquement de l'espace lorsque votre hôte supprime des données.

Exemples

Voir les exemples ci-dessous :

- Création d'un volume de 10 Gio :

```
docker volume create -d netapp --name demo -o size=10G -o
encryption=true
```

- Créez un volume de 100 Gio avec les snapshots :

```
docker volume create -d netapp --name demo -o size=100G -o
snapshotPolicy=default -o snapshotReserve=10
```

- Créez un volume dont le bit setuid est activé :

```
docker volume create -d netapp --name demo -o unixPermissions=4755
```

La taille minimale du volume est de 20MiB.

Si la réserve Snapshot n'est pas spécifiée et que la règle Snapshot est `none`, Trident utilise une réserve Snapshot de 0 %.

- Créer un volume sans policy de snapshots et sans réserve de snapshots :

```
docker volume create -d netapp --name my_vol --opt snapshotPolicy=none
```

- Créer un volume sans policy snapshot et une réserve Snapshot personnalisée de 10 % :

```
docker volume create -d netapp --name my_vol --opt snapshotPolicy=none
--opt snapshotReserve=10
```

- Créer un volume avec une règle Snapshot et une réserve Snapshot personnalisée de 10 % :

```
docker volume create -d netapp --name my_vol --opt
snapshotPolicy=myPolicy --opt snapshotReserve=10
```

- Créer un volume avec une règle Snapshot et accepter la réserve Snapshot par défaut d'ONTAP (généralement 5 %) :

```
docker volume create -d netapp --name my_vol --opt
snapshotPolicy=myPolicy
```

Options de volumes du logiciel Element

Les options du logiciel Element présentent les règles de taille et de qualité de services associées au volume. Lorsque le volume est créé, la politique de QoS associée à celui-ci est spécifiée à l'aide du `-o type=service_level` nomenclature

La première étape pour définir un niveau de service QoS avec le pilote Element consiste à créer au moins un type et à spécifier les IOPS minimum, maximum et en rafale associées à un nom dans le fichier de configuration.

Les autres options de création de volumes du logiciel Element sont les suivantes :

Option	Description
`size`	La taille du volume, par défaut 1Gio ou entrée de configuration … "Par défaut": {"size": "5G"}.
`blocksize`	Utilisez 512 ou 4096, par défaut 512 ou l'entrée de configuration DefaultBlockSize.

Exemple

Voir l'exemple de fichier de configuration suivant avec les définitions QoS :

```
{
    "...": "..."
    "Types": [
        {
            "Type": "Bronze",
            "Qos": {
                "minIOPS": 1000,
                "maxIOPS": 2000,
                "burstIOPS": 4000
            }
        },
        {
            "Type": "Silver",
            "Qos": {
                "minIOPS": 4000,
                "maxIOPS": 6000,
                "burstIOPS": 8000
            }
        },
        {
            "Type": "Gold",
            "Qos": {
                "minIOPS": 6000,
                "maxIOPS": 8000,
                "burstIOPS": 10000
            }
        }
    ]
}
```

Dans la configuration ci-dessus, nous avons trois définitions de règles : bronze, Silver et Gold. Ces noms sont arbitraires.

- Création d'un volume Gold de 10 Gio :

```
docker volume create -d solidfire --name sfGold -o type=Gold -o size=10G
```

- Créez un volume Bronze de 100 Gio :

```
docker volume create -d solidfire --name sfBronze -o type=Bronze -o
size=100G
```

Collecte des journaux

Vous pouvez recueillir des journaux pour obtenir de l'aide en matière de dépannage. La méthode que vous utilisez pour collecter les journaux varie en fonction de l'exécution du plug-in Docker.

Collecte des journaux pour le dépannage

Étapes

1. Si vous exécutez Astra Trident à l'aide de la méthode de plug-in géré recommandée (par exemple, à l'aide de `docker plugin` les commandes), les afficher comme suit :

```
docker plugin ls
ID                  NAME                DESCRIPTION
ENABLED
4fb97d2b956b        netapp:latest       nDVP - NetApp Docker Volume
Plugin   false
journalctl -u docker | grep 4fb97d2b956b
```

   Le niveau d'enregistrement standard devrait vous permettre de diagnostiquer la plupart des problèmes. Si cela ne suffit pas, vous pouvez activer la journalisation de débogage.

2. Pour activer la journalisation de débogage, installez le plug-in avec la journalisation de débogage activée :

```
docker plugin install netapp/trident-plugin:<version> --alias <alias>
debug=true
```

   Ou activez la journalisation de débogage lorsque le plug-in est déjà installé :

```
docker plugin disable <plugin>
docker plugin set <plugin> debug=true
docker plugin enable <plugin>
```

3. Si vous exécutez le binaire lui-même sur l'hôte, les journaux sont disponibles dans celui-ci `/var/log/netappdvp` répertoire. Pour activer la journalisation de débogage, spécifiez `-debug` lorsque vous exécutez le plug-in.

Conseils généraux de dépannage

- Le problème le plus courant auquel les nouveaux utilisateurs se sont exécutés est une mauvaise configuration qui empêche le plug-in de s'initialiser. Lorsque cela se produit, vous verrez probablement un message tel que celui-ci lorsque vous essayez d'installer ou d'activer le plug-in :

  ```
  Error response from daemon: dial unix /run/docker/plugins/<id>/netapp.sock:
  connect: no such file or directory
  ```

  Cela signifie que le plug-in n'a pas démarré. Heureusement, le plug-in a été conçu avec une fonctionnalité de journalisation complète qui devrait vous aider à diagnostiquer la plupart des problèmes que vous êtes susceptible de venir.

- En cas de problème de montage d'un PV sur un conteneur, vérifiez que `rpcbind` est installé et en cours d'exécution. Utilisez le gestionnaire de packages requis pour le système d'exploitation hôte et vérifiez si `rpcbind` est en cours d'exécution. Vous pouvez vérifier l'état du service rpcbind en exécutant un `systemctl status rpcbind` ou son équivalent.

Gérez plusieurs instances Trident d'Astra

Vous avez besoin de plusieurs instances de Trident lorsque vous souhaitez disposer de plusieurs configurations de stockage simultanément. La clé pour plusieurs instances est de leur donner des noms différents à l'aide de `--alias` avec le plug-in conteneurisé, ou `--volume-driver` Option lors de l'instanciation de Trident sur l'hôte.

Étapes du plug-in géré par Docker (version 1.13/17.03 ou ultérieure)

1. Lancez la première instance en spécifiant un alias et un fichier de configuration.

   ```
   docker plugin install --grant-all-permissions --alias silver
   netapp/trident-plugin:21.07 config=silver.json
   ```

2. Lancez la deuxième instance, en spécifiant un autre alias et un fichier de configuration.

   ```
   docker plugin install --grant-all-permissions --alias gold
   netapp/trident-plugin:21.07 config=gold.json
   ```

3. Créez des volumes spécifiant l'alias comme nom de pilote.

   Par exemple, pour le volume Gold :

   ```
   docker volume create -d gold --name ntapGold
   ```

Par exemple, pour le volume Silver :

```
docker volume create -d silver --name ntapSilver
```

Étapes pour les versions traditionnelles (version 1.12 ou antérieure)

1. Lancez le plug-in avec une configuration NFS à l'aide d'un ID de pilote personnalisé :

```
sudo trident --volume-driver=netapp-nas --config=/path/to/config
-nfs.json
```

2. Lancez le plug-in avec une configuration iSCSI à l'aide d'un ID de pilote personnalisé :

```
sudo trident --volume-driver=netapp-san --config=/path/to/config
-iscsi.json
```

3. Provisionnement de volumes Docker pour chaque instance de pilote :

   Par exemple pour NFS :

```
docker volume create -d netapp-nas --name my_nfs_vol
```

   Par exemple pour iSCSI :

```
docker volume create -d netapp-san --name my_iscsi_vol
```

Options de configuration du stockage

Découvrez les options de configuration disponibles pour vos configurations Astra Trident.

Options de configuration globale

Ces options de configuration s'appliquent à toutes les configurations Astra Trident, quelle que soit la plateforme de stockage utilisée.

Option	Description	Exemple
`version`	Numéro de version du fichier de configuration	`1`

Option	Description	Exemple
`storageDriverName`	Nom du pilote de stockage	`ontap-nas`, `ontap-san`, `ontap-nas-economy`, `ontap-nas-flexgroup`, `solidfire-san`
`storagePrefix`	Préfixe facultatif pour les noms de volumes. Valeur par défaut : `netappdvp_`.	`staging_`
`limitVolumeSize`	Restriction facultative sur les tailles de volume. Par défaut : « » (non appliqué)	`10g`

Ne pas utiliser `storagePrefix` (Y compris la valeur par défaut) pour les systèmes back-end Element. Par défaut, le `solidfire-san` le pilote ignore ce paramètre et n'utilise pas de préfixe. Nous vous recommandons d'utiliser un ID de tentID spécifique pour le mappage de volume Docker ou les données d'attributs renseignées par la version de Docker, les informations relatives au pilote et le nom brut de Docker dans les cas où il est possible d'utiliser une mundening de nom.

Les options par défaut sont disponibles pour éviter d'avoir à les spécifier sur chaque volume que vous créez. Le `size` option disponible pour tous les types de contrôleurs. Pour un exemple de définition de la taille de volume par défaut, reportez-vous à la section ONTAP configuration.

Option	Description	Exemple
`size`	Taille par défaut facultative pour les nouveaux volumes. Valeur par défaut : `1G`	`10G`

Configuration ONTAP

Outre les valeurs de configuration globale ci-dessus, lorsque vous utilisez ONTAP, les options de premier niveau suivantes sont disponibles.

Option	Description	Exemple
`managementLIF`	Adresse IP de la LIF de management ONTAP. Vous pouvez spécifier un nom de domaine complet (FQDN).	`10.0.0.1`

Option	Description	Exemple
`dataLIF`	Adresse IP de la LIF de protocole. **Pilotes NAS ONTAP**: Nous vous recommandons de spécifier `dataLIF`. Si elle n'est pas fournie, Astra Trident extrait les LIF de données du SVM. Vous pouvez spécifier un nom de domaine complet (FQDN) à utiliser pour les opérations de montage NFS, permettant de créer un DNS Round-Robin pour équilibrer la charge sur plusieurs LIF de données. **Pilotes SAN ONTAP** : ne pas spécifier pour iSCSI. Astra Trident utilise "Mappage de LUN sélectif ONTAP" Pour découvrir les LIFs iSCI nécessaires à l'établissement d'une session multi-chemin. Un avertissement est généré si `dataLIF` est explicitement défini.	`10.0.0.2`
`svm`	Storage Virtual machine à utiliser (requis, si la LIF de gestion est une LIF de cluster)	`svm_nfs`
`username`	Nom d'utilisateur pour la connexion au périphérique de stockage	`vsadmin`
`password`	Mot de passe pour se connecter au périphérique de stockage	`secret`
`aggregate`	Agrégat pour le provisionnement (facultatif ; si défini, doit être attribué au SVM) Pour le `ontap-nas-flexgroup` pilote, cette option est ignorée. Tous les agrégats affectés à un SVM sont utilisés pour provisionner un volume FlexGroup.	`aggr1`
`limitAggregateUsage`	Facultatif, le provisionnement échoue si l'utilisation est supérieure à ce pourcentage	`75%`

Option	Description	Exemple
`nfsMountOptions`	Contrôle granulaire des options de montage NFS ; par défaut : «-o nfssvers=3 ». **Disponible uniquement pour le `ontap-nas` et `ontap-nas-economy` pilotes**. "Pour plus d'informations sur la configuration de l'hôte NFS, consultez ici".	`-o nfsvers=4`
`igroupName`	Le groupe initiateur utilisé par le plug-in ; par défaut, est `netappdvp`. **Disponible uniquement pour le `ontap-san` driver**.	`myigroup`
`limitVolumeSize`	Taille maximale du volume requesable et taille du volume parent qtree. **Pour le `ontap-nas-economy` Driver, cette option limite en outre la taille des volumes FlexVol qu'elle crée**.	`300g`
`qtreesPerFlexvol`	Le nombre maximal de qtrees par FlexVol doit être compris dans la plage [50, 300], la valeur par défaut est 200. **Pour le `ontap-nas-economy` Pilote, cette option permet de personnaliser le nombre maximal de qtrees par FlexVol**.	`300`

Les options par défaut sont disponibles pour éviter d'avoir à les spécifier sur chaque volume que vous créez :

Option	Description	Exemple
`spaceReserve`	Mode de réservation d'espace ; `none` (provisionnement fin) ou `volume` (épais)	`none`
`snapshotPolicy`	Règle Snapshot à utiliser ; la valeur par défaut est `none`	`none`
`snapshotReserve`	Pourcentage de réserve de snapshot. La valeur par défaut est « » pour accepter la valeur par défaut de ONTAP	`10`

Option	Description	Exemple
`splitOnClone`	Séparer un clone de son parent lors de sa création. Par défaut, la valeur est `false`	`false`
`encryption`	Active NetApp Volume Encryption (NVE) sur le nouveau volume ; valeur par défaut sur `false`. Pour utiliser cette option, NVE doit être sous licence et activé sur le cluster. Si NAE est activé sur le back-end, tous les volumes provisionnés dans Astra Trident seront activés par NAE. Pour plus d'informations, se reporter à : "Fonctionnement d'Astra Trident avec NVE et NAE".	vrai
`unixPermissions`	L'option NAS pour les volumes NFS provisionnés est définie par défaut sur `777`	`777`
`snapshotDir`	Option NAS pour accéder à l'`.snapshot` répertoire, par défaut sur `false`	`true`
`exportPolicy`	L'option NAS pour la export policy NFS à utiliser est définie par défaut sur `default`	`default`
`securityStyle`	Option NAS pour l'accès au volume NFS provisionné. Prise en charge de NFS `mixed` et `unix` styles de sécurité. La valeur par défaut est `unix`.	`unix`
`fileSystemType`	OPTION SAN pour sélectionner le type de système de fichiers, par défaut sur `ext4`	`xfs`
`tieringPolicy`	Règle de Tiering à utiliser ; la règle par défaut est `none`; `snapshot-only` Pour la configuration SVM-DR antérieure à ONTAP 9.5	`none`

Options d'évolutivité

Le `ontap-nas` et `ontap-san` Les pilotes créent un ONTAP FlexVol pour chaque volume Docker. ONTAP prend en charge jusqu'à 1000 volumes FlexVol par nœud de cluster avec un cluster maximum de 12,000 volumes FlexVol. Si votre volume Docker répond à cette restriction, le `ontap-nas` Le pilote est la solution NAS préférée du fait des fonctionnalités supplémentaires offertes par les volumes FlexVol, telles que les snapshots et le clonage granulaires avec volume Docker.

Si vous avez besoin de plus de volumes Docker que ne peut pas être pris en charge par les limites FlexVol, choisissez la `ontap-nas-economy` ou le `ontap-san-economy` conducteur.

Le `ontap-nas-economy` Le pilote crée des volumes Docker en tant que qtrees ONTAP dans un pool de volumes FlexVol gérés automatiquement. Les qtrees offrent une évolutivité largement supérieure, jusqu'à 100,000 par nœud de cluster et 2,400,000 par cluster, au détriment de certaines fonctionnalités. Le `ontap-nas-economy` Le pilote ne prend pas en charge le clonage ou les snapshots granulaires volume Docker.

> Le `ontap-nas-economy` Le pilote n'est pas pris en charge par Docker Swarm, car Swarm n'effectue pas la création de volumes entre plusieurs nœuds.

Le `ontap-san-economy` Le pilote crée des volumes Docker en tant que LUN ONTAP dans un pool partagé de volumes FlexVol gérés automatiquement. De cette façon, chaque FlexVol n'est pas limité à un seul LUN et offre une meilleure évolutivité pour les charges de travail SAN. Selon les baies de stockage, ONTAP prend en charge jusqu'à 16384 LUN par cluster. Comme les volumes sont sous-LUN, ce pilote prend en charge les snapshots et le clonage granulaires par volume Docker.

Choisissez le `ontap-nas-flexgroup` pilote pour augmenter le parallélisme vers un seul volume qui peut atteindre plusieurs pétaoctets avec des milliards de fichiers. Les utilisations idéales de FlexGroups sont l'IA, LE ML, le Big Data et l'analytique, les logiciels, le streaming, les référentiels de fichiers, etc. Trident utilise tous les agrégats attribués à un SVM lors du provisionnement d'un volume FlexGroup. La prise en charge d'FlexGroup dans Trident comporte également plusieurs considérations :

- Requiert ONTAP version 9.2 ou supérieure
- À ce jour, FlexGroups prend uniquement en charge NFS v3.
- Recommandé pour activer les identifiants NFSv3 64 bits pour la SVM.
- La taille minimale recommandée de FlexGroup est de 100 Go.
- Le clonage n'est pas pris en charge pour FlexGroup volumes.

Pour plus d'informations sur FlexGroups et les workloads appropriés à FlexGroups, consultez le "NetApp FlexGroup Volume Guide des meilleures pratiques et de mise en œuvre".

Pour bénéficier de fonctionnalités avancées et d'une évolutivité massive dans le même environnement, vous pouvez exécuter plusieurs instances du plug-in de volume Docker, en utilisant une seule instance `ontap-nas` et une autre utilisation `ontap-nas-economy`.

Exemples de fichiers de configuration ONTAP

Exemple NFS pour `ontap-nas` pilote

```
{
    "version": 1,
    "storageDriverName": "ontap-nas",
    "managementLIF": "10.0.0.1",
    "dataLIF": "10.0.0.2",
    "svm": "svm_nfs",
    "username": "vsadmin",
    "password": "password",
    "aggregate": "aggr1",
    "defaults": {
      "size": "10G",
      "spaceReserve": "none",
      "exportPolicy": "default"
    }
}
```

Exemple NFS pour `ontap-nas-flexgroup` pilote

```
{
    "version": 1,
    "storageDriverName": "ontap-nas-flexgroup",
    "managementLIF": "10.0.0.1",
    "dataLIF": "10.0.0.2",
    "svm": "svm_nfs",
    "username": "vsadmin",
    "password": "password",
    "defaults": {
      "size": "100G",
      "spaceReserve": "none",
      "exportPolicy": "default"
    }
}
```

Exemple NFS pour `ontap-nas-economy` pilote

```
{
    "version": 1,
    "storageDriverName": "ontap-nas-economy",
    "managementLIF": "10.0.0.1",
    "dataLIF": "10.0.0.2",
    "svm": "svm_nfs",
    "username": "vsadmin",
    "password": "password",
    "aggregate": "aggr1"
}
```

Exemple iSCSI pour `ontap-san` pilote

```
{
    "version": 1,
    "storageDriverName": "ontap-san",
    "managementLIF": "10.0.0.1",
    "dataLIF": "10.0.0.3",
    "svm": "svm_iscsi",
    "username": "vsadmin",
    "password": "password",
    "aggregate": "aggr1",
    "igroupName": "myigroup"
}
```

Exemple NFS pour `ontap-san-economy` pilote

```
{
    "version": 1,
    "storageDriverName": "ontap-san-economy",
    "managementLIF": "10.0.0.1",
    "dataLIF": "10.0.0.3",
    "svm": "svm_iscsi_eco",
    "username": "vsadmin",
    "password": "password",
    "aggregate": "aggr1",
    "igroupName": "myigroup"
}
```

Configuration logicielle Element

Outre les valeurs de configuration globale, lorsque le logiciel Element (NetApp HCI/SolidFire) est utilisé, ces options sont disponibles.

Option	Description	Exemple
`Endpoint`	https://<login>:<password>@<mvip>/json-rpc/<element-version>	https://admin:admin@192.168.160.3/json-rpc/8.0
`SVIP`	Port et adresse IP iSCSI	10.0.0.7:3260
`TenantName`	Locataire SolidFireF à utiliser (créé s'il n'est pas trouvé)	`docker`
`InitiatorIFace`	Spécifiez l'interface lors de la restriction du trafic iSCSI à une interface non-par défaut	`default`
`Types`	Spécifications de QoS	Voir l'exemple ci-dessous
`LegacyNamePrefix`	Préfixe des installations Trident mises à niveau. Si vous avez utilisé une version de Trident antérieure à 1.3.2 et que vous effectuez une mise à niveau avec des volumes existants, vous devez définir cette valeur pour accéder à vos anciens volumes mappés via la méthode nom-volume.	`netappdvp-`

Le `solidfire-san` Le pilote ne prend pas en charge Docker Swarm.

Exemple de fichier de configuration du logiciel Element

```
{
    "version": 1,
    "storageDriverName": "solidfire-san",
    "Endpoint": "https://admin:admin@192.168.160.3/json-rpc/8.0",
    "SVIP": "10.0.0.7:3260",
    "TenantName": "docker",
    "InitiatorIFace": "default",
    "Types": [
        {
            "Type": "Bronze",
            "Qos": {
                "minIOPS": 1000,
                "maxIOPS": 2000,
                "burstIOPS": 4000
            }
        },
        {
            "Type": "Silver",
            "Qos": {
                "minIOPS": 4000,
                "maxIOPS": 6000,
                "burstIOPS": 8000
            }
        },
        {
            "Type": "Gold",
            "Qos": {
                "minIOPS": 6000,
                "maxIOPS": 8000,
                "burstIOPS": 10000
            }
        }
    ]
}
```

Problèmes et limites connus

Découvrez des informations sur les problèmes et les limites connus avec Astra Trident avec Docker.

La mise à niveau de Trident Docker Volume Plug-in vers la version 20.10 et ultérieure à partir des versions plus anciennes entraîne un échec de mise à niveau, sans erreur de fichier ou de répertoire de ce type.

Solution de contournement

1. Désactivez le plug-in.

```
docker plugin disable -f netapp:latest
```

2. Retirez le plug-in.

```
docker plugin rm -f netapp:latest
```

3. Réinstallez le plug-in en fournissant le complément `config` paramètre.

```
docker plugin install netapp/trident-plugin:20.10 --alias netapp --grant
-all-permissions config=config.json
```

Les noms de volumes doivent comporter au moins 2 caractères.

Il s'agit d'une limitation client Docker. Le client interprète un seul nom de caractère comme étant un chemin Windows. "Voir bug 25773".

Docker Swarm dispose de certains comportements qui empêchent Astra Trident de les prendre en charge avec chaque combinaison de stockage et de pilote.

- Docker Swarm utilise actuellement le nom du volume, mais pas l'ID de volume, comme identifiant de volume unique.
- Les requêtes de volume sont envoyées simultanément à chaque nœud d'un cluster Swarm.
- Les plug-ins de volumes (y compris Astra Trident) doivent être exécutés de manière indépendante sur chaque nœud d'un cluster Swarm. Grâce au fonctionnement de ONTAP et à son mode de fonctionnement `ontap-nas` et `ontap-san` le conducteur fonctionne, ce sont les seuls qui peuvent être capables de fonctionner dans ces limites.

Les autres conducteurs sont sujets à des problèmes tels que les conditions de course qui peuvent entraîner la création d'un grand nombre de volumes pour une seule demande sans un « gagnant » clair ; par exemple, l'élément possède une fonctionnalité qui permet aux volumes d'avoir le même nom mais des ID différents.

NetApp a fourni des commentaires à l'équipe Docker, mais ne fournit aucune indication de recours futur.

Si un FlexGroup est provisionné, ONTAP ne provisionne pas un deuxième FlexGroup si le deuxième FlexGroup dispose d'un ou de plusieurs agrégats en commun avec la FlexGroup provisionnée.

Foire aux questions

Trouvez les réponses aux questions fréquemment posées concernant l'installation, la configuration, la mise à niveau et le dépannage d'Astra Trident.

Questions générales

À quelle fréquence Astra Trident est-il commercialisé ?

Astra Trident est commercialisé tous les trois mois : janvier, avril, juillet et octobre. Ceci fait un mois après une version de Kubernetes.

Est-ce que l'Astra Trident prend en charge toutes les fonctionnalités publiées dans une version particulière de Kubernetes ?

Astra Trident ne prend généralement pas en charge les fonctionnalités alpha dans Kubernetes. Trident peut prendre en charge les fonctionnalités bêta dans les deux versions de Trident, qui suivent la version bêta de Kubernetes.

Est-ce que l'Astra Trident se base sur d'autres produits NetApp pour son fonctionnement ?

Astra Trident ne fonctionne pas sous forme de dépendance vis-à-vis d'autres logiciels NetApp, et il fonctionne comme une application autonome. Toutefois, vous devez disposer d'un système de stockage back-end NetApp.

Comment obtenir les informations complètes sur la configuration d'Astra Trident ?

Utilisez le `tridentctl get` Commande pour obtenir plus d'informations sur la configuration d'Astra Trident.

Puis-je obtenir des metrics sur le provisionnement du stockage par Astra Trident ?

Oui. Trident 20.01 introduit les terminaux Prometheus qui peuvent être utilisés pour collecter des informations sur le fonctionnement d'Astra Trident, notamment le nombre de systèmes back-end gérés, le nombre de volumes provisionnés, les octets consommés, etc. Vous pouvez également utiliser Cloud Insights pour la surveillance et l'analyse.

L'expérience utilisateur change-t-elle lors de l'utilisation d'Astra Trident en tant que provisionnement CSI ?

Non Il n'y a aucun changement en ce qui concerne l'expérience utilisateur et les fonctionnalités. Le nom de provisionnement utilisé est `csi.trident.netapp.io`. Cette méthode d'installation d'Astra Trident est recommandée si vous souhaitez utiliser toutes les nouvelles fonctionnalités fournies par les versions actuelles et futures.

Installez et utilisez Astra Trident sur un cluster Kubernetes

Quelles sont les versions prises en charge de `etcd`?

Astra Trident n'a plus besoin d'un `etcd`. Il utilise des CRD pour maintenir l'état.

Est-ce que Astra Trident prend en charge une installation hors ligne à partir d'un registre privé ?

Oui, Astra Trident peut être installé hors ligne. Voir "ici".

Puis-je installer Astra Trident à distance ?

Oui. Avec Astra Trident 18.10 et les versions ultérieures, il prend en charge la fonctionnalité d'installation à distance à partir de n'importe quelle machine `kubectl` accès au cluster. Après `kubectl` vérification de l'accès (par exemple, lancement d'un `kubectl get nodes` commande de la machine à distance pour la vérification), suivre les instructions d'installation.

Puis-je configurer la haute disponibilité avec Astra Trident ?

Astra Trident est installé en tant que Kubernetes Deployment (ReplicaSet) avec une instance, et ce de même qu'il a intégré la haute disponibilité. Vous ne devez pas augmenter le nombre de répliques dans le déploiement. Si le nœud sur lequel Astra Trident est installé ou si le pod est inaccessible, Kubernetes redéploie automatiquement le pod sur un nœud en état de santé dans votre cluster. L'Astra Trident est uniquement du plan de contrôle. Les pods actuellement montés ne sont donc pas affectés en cas de redéploiement de l'Astra Trident.

Astra Trident a-t-il besoin d'accéder au namespace du système kube ?

L'Astra Trident effectue une lecture sur le serveur d'API Kubernetes afin de déterminer le moment où les applications demandent des nouveaux ESV. Il doit donc accéder à kube-System.

Quels sont les rôles et privilèges utilisés par Astra Trident ?

Le programme d'installation de Trident crée un cluster Kubernetes ClusterRole qui dispose d'un accès spécifique au volume persistant, à la demande de volume persistant, à la classe de stockage et aux ressources secrètes du cluster Kubernetes. Voir "ici".

Est-il possible de générer localement les fichiers de manifeste exacts qu'Astra Trident utilise pour l'installation ?

Vous pouvez générer et modifier localement les fichiers de manifeste exacts utilisés par Astra Trident pour l'installation, si nécessaire. Voir "ici".

Puis-je partager le même SVM back-end ONTAP pour deux instances Astra Trident distinctes pour deux clusters Kubernetes distincts ?

Bien qu'il ne soit pas conseillé, vous pouvez utiliser le même SVM back-end pour deux instances Astra Trident. Spécifiez un nom de volume unique pour chaque instance lors de l'installation et/ou spécifiez un nom unique `StoragePrefix` paramètre dans le `setup/backend.json` fichier. Ceci permet de s'assurer que le même FlexVol n'est pas utilisé pour les deux instances.

Est-il possible d'installer Astra Trident sous ContainerLinux (anciennement CoreOS) ?

Astra Trident est un simple pod Kubernetes. Il peut être installé quel que soit l'emplacement de Kubernetes.

Puis-je utiliser Astra Trident avec NetApp Cloud Volumes ONTAP ?

Oui, Astra Trident est pris en charge par AWS, Google Cloud et Azure.

Astra Trident fonctionne-t-il avec NetApp Cloud volumes Services ?

Oui. Astra Trident prend en charge le service Azure NetApp Files dans Azure ainsi que le service Cloud Volumes Service dans GCP.

Dépannage et support

NetApp prend-il en charge Astra Trident ?

Bien qu'Astra Trident soit open source et fourni gratuitement, NetApp le prend entièrement en charge à condition que votre système back-end NetApp soit pris en charge.

Comment puis-je soulever un dossier de demande de support ?

Pour soulever un dossier de support, effectuez l'une des opérations suivantes :

1. Contactez votre support Account Manager pour obtenir de l'aide pour créer un dossier.
2. Pour ouvrir un dossier de demande de support, contactez "Support NetApp".

Comment générer un bundle de journaux de support ?

Vous pouvez créer un bundle de support en exécutant `tridentctl logs -a`. Outre les journaux capturés dans le pack, capture le journal kubelet pour diagnostiquer les problèmes de montage côté Kubernetes. Les instructions d'obtention du journal kubelet varient en fonction de l'installation de Kubernetes.

Que faire si j'ai besoin de demander une nouvelle fonctionnalité ?

Créer un problème sur "Astra Trident Github" Et mentionner **RFE** dans le sujet et la description du problème.

Où puis-je soulever un défaut ?

Créer un problème sur "Astra Trident Github". Veillez à inclure toutes les informations et tous les journaux nécessaires concernant le problème.

Que se passe-t-il si j'ai une brève question sur Astra Trident et que j'ai besoin de précisions ? Y a-t-il une communauté ou un forum?

Si vous avez des questions, des problèmes ou des demandes, contactez-nous par le biais de notre Astra "Déroulez le canal" Ou GitHub.

Le mot de passe de mon système de stockage a changé et Astra Trident ne fonctionne plus, comment récupérer?

Mettez à jour le mot de passe du backend avec `tridentctl update backend myBackend -f </path/to_new_backend.json> -n trident`. Remplacement `myBackend` dans l'exemple avec votre nom de back-end, et `/path/to_new_backend.json` avec le chemin d'accès correct `backend.json` fichier.

Astra Trident ne trouve pas le nœud Kubernetes. Comment résoudre ce problème ?

Il existe deux scénarios possibles pour savoir pourquoi Astra Trident ne peut pas trouver un nœud Kubernetes. Elle peut être due à un problème de mise en réseau dans Kubernetes ou DNS. Le demonset de nœuds Trident qui s'exécute sur chaque nœud Kubernetes doit pouvoir communiquer avec le contrôleur Trident pour enregistrer le nœud avec Trident. Si des modifications de mise en réseau ont eu lieu après l'installation d'Astra Trident, ce problème se produit uniquement avec les nouveaux nœuds Kubernetes qui sont ajoutés au cluster.

Si le pod Trident est détruit, ces données seront-elles perdues ?

Les données ne seront pas perdues si le pod Trident est détruit. Les métadonnées de Trident sont stockées dans des objets CRD. Tous les volumes persistants provisionnés par Trident fonctionneront normalement.

Mettez à niveau Astra Trident

Est-il possible de mettre à niveau une version plus ancienne directement vers une version plus récente (sans passer par quelques versions) ?

NetApp prend en charge la mise à niveau d'Astra Trident d'une version majeure à la prochaine version majeure. Vous pouvez effectuer la mise à niveau de la version 18.xx vers la version 19.xx, 19.xx vers la version 20.xx, etc. Il est conseillé de tester la mise à niveau dans un laboratoire avant le déploiement en production.

Est-il possible de revenir à une version antérieure de Trident ?

Un certain nombre de facteurs doivent être évalués si vous voulez rétrograder. Voir "la section sur la rétrogradation".

Gestion des systèmes back-end et des volumes

Dois-je définir à la fois des LIF de données et de gestion dans un fichier de définition du back-end ONTAP ?

Le LIF de gestion est obligatoire. Data LIF varie :

- San ONTAP : ne spécifiez pas pour iSCSI. Astra Trident utilise "Mappage de LUN sélectif ONTAP" Pour découvrir les LIFs iSCI nécessaires à l'établissement d'une session multi-chemin. Un avertissement est généré si `dataLIF` est explicitement défini. Reportez-vous à la section "Options et exemples de configuration des SAN ONTAP" pour plus d'informations.
- NAS ONTAP : spécification recommandée `dataLIF`. Si elle n'est pas fournie, Astra Trident extrait les LIF de données du SVM. Vous pouvez spécifier un nom de domaine complet (FQDN) à utiliser pour les opérations de montage NFS, permettant de créer un DNS Round-Robin pour équilibrer la charge sur plusieurs LIF de données. Reportez-vous à la section "Options et exemples de configuration du NAS

ONTAP" pour plus d'informations

L'Astra Trident peut-il configurer le protocole CHAP pour les systèmes back-end ONTAP ?

Oui. À partir de la version 20.04, Astra Trident prend en charge le protocole CHAP bidirectionnel pour les systèmes back-end ONTAP. Ceci nécessite un paramètre `useCHAP=true` dans votre configuration back-end.

Comment gérer les règles d'exportation avec Astra Trident ?

Astra Trident peut créer et gérer de manière dynamique des règles d'exportation à partir de la version 20.04. Cela permet à l'administrateur de stockage de fournir un ou plusieurs blocs CIDR dans leur configuration backend et de laisser Trident ajouter des adresses IP de nœud comprise dans ces plages à une export policy créée. Ainsi, Astra Trident gère automatiquement l'ajout et la suppression de règles pour les nœuds dont les adresses IP sont comprises dans les rapports CIDR donnés. Cette fonctionnalité requiert CSI Trident.

Pouvons-nous spécifier un port dans le DataLIF ?

Astra Trident 19.01 et les versions ultérieures prennent en charge la spécification d'un port dans le DataLIF. Configurez-le dans le `backend.json` fichier en tant que `"managementLIF": <ip address>:<port>"`. Par exemple, si l'adresse IP de votre LIF de gestion est 192.0.2.1, et si le port est 1000, configurez `"managementLIF": "192.0.2.1:1000"`.

Les adresses IPv6 peuvent-elles être utilisées pour les LIF de données et de gestion ?

Astra Trident prend en charge la définition des adresses IPv6 pour :

- `managementLIF` et `dataLIF` Pour les systèmes NAS ONTAP.
- `managementLIF` Pour les systèmes back-end ONTAP SAN. Vous ne pouvez pas spécifier `dataLIF` Sur un SAN backend ONTAP.

Astra Trident doit être installé à l'aide du `--use-ipv6` Indicateur pour qu'il fonctionne sur IPv6.

Est-il possible de mettre à jour la LIF de gestion en back-end ?

Oui, il est possible de mettre à jour la LIF de management back-end à l'aide de `tridentctl update backend` commande.

Est-il possible de mettre à jour la LIF de données sur le backend ?

Vous pouvez mettre à jour la LIF de données sur `ontap-nas` et `ontap-nas-economy` uniquement.

Est-il possible de créer plusieurs systèmes back-end dans Astra Trident pour Kubernetes ?

Astra Trident peut prendre en charge de nombreux systèmes back-end simultanément, avec le même pilote ou des pilotes différents.

Comment Astra Trident stocke-t-il les identifiants back-end ?

Astra Trident stocke les identifiants back-end sous le titre de secrets de Kubernetes.

Comment l'Astra Trident sélectionne-t-il un système back-end spécifique ?

Si les attributs back-end ne peuvent pas être utilisés pour sélectionner automatiquement les pools appropriés pour une classe, l' `storagePools` et `additionalStoragePools` les paramètres sont utilisés pour sélectionner un ensemble spécifique de pools.

Comment s'assurer qu'Astra Trident ne provisionne pas d'un back-end spécifique ?

Le `excludeStoragePools` Paramètre utilisé pour filtrer l'ensemble de pools qu'Astra Trident utilisera pour le provisionnement et supprimera tous les pools correspondant.

Si plusieurs systèmes back-end sont de même type, comment Astra Trident sélectionne-il le back-end à utiliser ?

Si plusieurs systèmes back-end configurés du même type sont configurés, Astra Trident sélectionne le back-end approprié en fonction des paramètres présents dans `StorageClass` et `PersistentVolumeClaim`. Par exemple, si il existe plusieurs pilotes back-end ontap-nas, Astra Trident tente de correspondre aux paramètres dans le `StorageClass` et `PersistentVolumeClaim` combiné et correspondre à un système back-end capable de fournir les exigences répertoriées dans `StorageClass` et `PersistentVolumeClaim`. Si plusieurs systèmes back-end correspondent à la demande, l'Astra Trident est sélectionnée de manière aléatoire.

Astra Trident prend-il en charge le protocole CHAP bidirectionnel avec Element/SolidFire ?

Oui.

Comment Astra Trident déploie-t-il des qtrees sur un volume ONTAP ? Combien de qtrees peuvent-ils être déployés sur un seul volume ?

Le `ontap-nas-economy` Le pilote crée jusqu'à 200 qtrees dans le même FlexVol (configurables entre 50 et 300), 100,000 qtrees par nœud de cluster et 2,4 millions par cluster. Lorsque vous saisissez un nouveau `PersistentVolumeClaim` Le pilote cherche à voir si un FlexVol existe déjà pour le service du nouveau qtree. Si la FlexVol n'existe pas qui peut traiter le qtree, un nouveau FlexVol est créé.

Comment définir des autorisations Unix pour les volumes provisionnés sur ONTAP NAS ?

Vous pouvez définir des autorisations Unix sur le volume provisionné par Astra Trident en définissant un paramètre dans le fichier de définition backend.

Comment configurer un ensemble explicite d'options de montage NFS ONTAP lors du provisionnement d'un volume ?

Par défaut, Astra Trident ne définit pas d'option de montage sur aucune valeur avec Kubernetes. Pour spécifier les options de montage dans la classe de stockage Kubernetes, suivez l'exemple donné "ici".

Comment définir les volumes provisionnés sur une export policy spécifique ?

Pour permettre aux hôtes appropriés d'accéder à un volume, utilisez le `exportPolicy` paramètre configuré dans le fichier de définition backend.

Comment définir le chiffrement de volume avec Astra Trident et ONTAP ?

Vous pouvez définir le chiffrement sur le volume provisionné par Trident à l'aide du paramètre de chiffrement dans le fichier de définition back-end. Pour plus d'informations, se reporter à : "Fonctionnement d'Astra Trident avec NVE et NAE"

Quelle est la meilleure façon d'implémenter la QoS pour ONTAP avec Astra Trident ?

Utiliser `StorageClasses` Afin d'implémenter la QoS pour ONTAP.

Comment puis-je spécifier le provisionnement fin ou non fin avec Astra Trident ?

Les pilotes ONTAP prennent en charge le provisionnement fin ou non fin. Le provisionnement fin est par défaut pour les pilotes ONTAP. Si un provisionnement lourd est souhaité, vous devez configurer le fichier de définition backend ou le `StorageClass`. Si les deux sont configurés, `StorageClass` a priorité. Configurez les éléments suivants pour ONTAP :

1. Marche `StorageClass`, réglez le `provisioningType` attribuer comme épaisseur.
2. Dans le fichier de définition back-end, activez les volumes épais par définition `backend spaceReserve parameter` comme volume.

Comment puis-je m'assurer que les volumes utilisés ne sont pas supprimés même si je supprime accidentellement le volume de volume persistant ?

La protection contre la demande de volume persistant est automatiquement activée sur Kubernetes à partir de la version 1.10.

Puis-je augmenter les demandes de volume persistant NFS créées par Astra Trident ?

Oui. Vous pouvez développer un volume de volume persistant créé par Astra Trident. Notez que la croissance automatique de volume est une fonctionnalité ONTAP qui n'est pas applicable à Trident.

Si je dispose d'un volume créé en dehors d'Astra Trident, est-il possible de l'importer dans Astra Trident ?

À partir de la version 19.04, vous pouvez utiliser la fonctionnalité d'importation de volumes pour intégrer des volumes dans Kubernetes.

Puis-je importer un volume en mode SnapMirror Data protection (DP) ou hors ligne ?

L'importation du volume échoue si le volume externe est en mode DP ou est hors ligne. Vous recevez le message d'erreur suivant :

```
Error: could not import volume: volume import failed to get size of
volume: volume <name> was not found (400 Bad Request) command terminated
with exit code 1.
Make sure to remove the DP mode or put the volume online before importing
the volume.
```

Puis-je étendre les demandes de volume persistant iSCSI créées par Astra Trident ?

Trident 19.10 prend en charge l'extension de volumes persistants iSCSI à l'aide du mécanisme de provisionnement CSI.

Comment un quota de ressources est-il traduit-il vers un cluster NetApp ?

Le quota de ressources de stockage Kubernetes doit fonctionner tant que le stockage NetApp possède de la capacité. Lorsque le stockage NetApp ne peut pas respecter les paramètres de quota Kubernetes en raison d'un manque de capacité, Astra Trident tente d'effectuer le provisionnement, mais s'y efforce d'erreurs.

Est-il possible de créer des copies Snapshot de volume avec Astra Trident ?

Oui. La création à la demande de copies Snapshot de volume et de volumes persistants à partir de copies Snapshot est prise en charge par Astra Trident. Pour créer des volumes persistants à partir de snapshots, assurez-vous que l' `VolumeSnapshotDataSource` la porte de fonction a été activée.

Quels sont les pilotes qui prennent en charge les copies Snapshot de volume Astra Trident ?

Depuis, nous proposons aujourd'hui la prise en charge de snapshots à la demande `ontap-nas`, `ontap-nas-flexgroup`, `ontap-san`, `ontap-san-economy`, `solidfire-san`, `gcp-cvs`, et `azure-netapp-files` pilotes backend.

Comment effectuer une sauvegarde Snapshot d'un volume provisionné par Astra Trident avec ONTAP ?

Cette option est disponible sur `ontap-nas`, `ontap-san`, et `ontap-nas-flexgroup` pilotes. Vous pouvez également spécifier un `snapshotPolicy` pour le `ontap-san-economy` Pilote au niveau FlexVol.

Cette fonction est également disponible sur le `ontap-nas-economy` Pilotes mais au niveau de la granularité FlexVol, pas au niveau de la granularité qtree. Pour permettre la création de copies Snapshot provisionnées par Astra Trident, définissez le paramètre back-end `snapshotPolicy` À la politique de snapshot souhaitée, telle que définie sur le back-end ONTAP. Tout snapshot effectué par le contrôleur de stockage ne est pas connu d'Astra Trident.

Puis-je définir un pourcentage de réserve de snapshot pour un volume provisionné via Astra Trident ?

Oui. Il est possible de réserver un pourcentage spécifique d'espace disque pour le stockage des copies Snapshot via Astra Trident en configurant le `snapshotReserve` attribut dans le fichier de définition backend. Si vous avez configuré `snapshotPolicy` et `snapshotReserve` dans le fichier de définition backend, le

pourcentage de réserve de snapshot est défini en fonction de la `snapshotReserve` pourcentage indiqué dans le fichier back-end. Si le `snapshotReserve` Le pourcentage de nombre n'est pas indiqué, ONTAP occupe par défaut le pourcentage de réserve Snapshot comme 5. Si le `snapshotPolicy` l'option est définie sur aucune, le pourcentage de réserve snapshot est défini sur 0.

Puis-je accéder directement au répertoire de snapshot de volume et copier les fichiers ?

Oui, vous pouvez accéder au répertoire de snapshots sur le volume provisionné par Trident en paramétrant le `snapshotDir` paramètre dans le fichier de définition backend.

Puis-je configurer SnapMirror pour des volumes avec Astra Trident ?

Actuellement, SnapMirror doit être défini en externe via l'interface de ligne de commande ONTAP ou OnCommand System Manager.

Comment restaurer des volumes persistants à un snapshot ONTAP spécifique ?

Pour restaurer un volume sur un snapshot ONTAP, effectuez les opérations suivantes :

1. Arrêter le pod d'application qui utilise le volume persistant.
2. Restaurez les données vers le snapshot requis via l'interface de ligne de commande de ONTAP ou OnCommand System Manager.
3. Redémarrez le pod d'application.

Trident peut-il provisionner des volumes sur des SVM dont un miroir de partage de charge est configuré ?

Des miroirs de partage de charge peuvent être créés pour les volumes root des SVM qui fournissent des données sur NFS. ONTAP met automatiquement à jour les miroirs de partage de charge pour les volumes qui ont été créés par Trident. Cela peut entraîner des retards dans le montage des volumes. Lorsque plusieurs volumes sont créés via Trident, le provisionnement d'un volume dépend de la mise à jour par ONTAP du miroir de partage de charge.

Comment puis-je séparer l'utilisation de la classe de stockage pour chaque client/locataire ?

Kubernetes n'autorise pas les classes de stockage dans les espaces de noms. Toutefois, vous pouvez utiliser Kubernetes pour limiter l'utilisation d'une classe de stockage spécifique par espace de noms à l'aide de quotas de ressources de stockage, qui sont par espace de noms. Pour refuser un accès d'espace de noms spécifique à un stockage spécifique, définissez le quota de ressources sur 0 pour cette classe de stockage.

Assistance

Astra Trident est un projet NetApp officiellement pris en charge. Vous pouvez contacter NetApp à l'aide de n'importe quel mécanisme standard et bénéficier d'un support haute performance adapté à vos besoins.

Il existe aussi une communauté publique très animée d'utilisateurs de conteneurs (y compris des développeurs Astra Trident) sur notre Astra "Déroulez le canal". C'est un endroit idéal pour poser des questions d'ordre général sur le projet et discuter de sujets connexes avec des pairs partageant des mêmes idées.

Dépannage

Utilisez les pointeurs indiqués ici pour résoudre les problèmes que vous pourriez rencontrer lors de l'installation et de l'utilisation d'Astra Trident.

Pour obtenir de l'aide avec Astra Trident, créez un bundle de support à l'aide de `tridentctl logs -a -n trident` et envoyez-le à `NetApp Support <Getting Help>`.

Pour obtenir une liste complète des articles de dépannage, reportez-vous au "Base de connaissances NetApp (identifiant requis)". Vous trouverez également des informations sur le dépannage des problèmes liés à Astra "ici".

Dépannage général

- Si le pod Trident ne fonctionne pas correctement (par exemple, lorsque le pod Trident est coincé dans le `ContainerCreating` phase avec moins de deux conteneurs prêts à l'emploi), en cours d'exécution `kubectl -n trident describe deployment trident` et `kubectl -n trident describe pod trident-**` peut fournir des informations exploitables supplémentaires. Obtenir des journaux kubelet (par exemple, via `journalctl -xeu kubelet`) peut également être utile.
- Si les journaux Trident ne contient pas suffisamment d'informations, vous pouvez essayer d'activer le mode de débogage pour Trident en passant le `-d` permet d'indiquer le paramètre d'installation en fonction de votre option d'installation.

  Vérifiez ensuite que le débogage est défini à l'aide de `./tridentctl logs -n trident` et à la recherche de `level=debug msg` dans le journal.

  Installé avec l'opérateur

  ```
  kubectl patch torc trident -n <namespace> --type=merge -p
  '{"spec":{"debug":true}}'
  ```

  Cela redémarrera tous les modules Trident, ce qui peut prendre plusieurs secondes. Vous pouvez le vérifier en observant la colonne « ÂGE » dans la sortie de `kubectl get pod -n trident`.

  Pour utilisation d'Astra Trident 20.07 et 20.10 `tprov` à la place de `torc`.

  Installé avec Helm

  ```
  helm upgrade <name> trident-operator-21.07.1-custom.tgz --set
  tridentDebug=true`
  ```

  Installé avec tridentctl

  ```
  ./tridentctl uninstall -n trident
  ./tridentctl install -d -n trident
  ```

- Vous pouvez également obtenir des journaux de débogage pour chaque back-end en incluant `debugTraceFlags` dans votre définition de back-end. Par exemple, incluez `debugTraceFlags: {"api":true, "method":true,}` Pour obtenir des appels d'API et des transits de méthode dans les journaux Trident. Systèmes back-end existants peuvent avoir lieu `debugTraceFlags` configuré avec un `tridentctl backend update`.
- Lorsque vous utilisez RedHat CoreOS, assurez-vous que cela `iscsid` est activé sur les nœuds workers et démarré par défaut. Pour ce faire, utilisez OpenShift MachineConfiguration ou modifiez les modèles d'allumage.
- Un problème courant que vous pouvez rencontrer avec Trident "Azure NetApp Files" lorsque les secrets de locataire et de client proviennent d'un enregistrement d'application avec des autorisations insuffisantes. Pour obtenir la liste complète des exigences Trident, reportez-vous à la section "Azure NetApp Files" configuration.
- En cas de problème de montage d'un PV sur un conteneur, vérifiez que `rpcbind` est installé et en cours d'exécution. Utilisez le gestionnaire de packages requis pour le système d'exploitation hôte et vérifiez si `rpcbind` est en cours d'exécution. Vous pouvez vérifier le statut de l' `rpcbind` service en exécutant un `systemctl status rpcbind` ou son équivalent.
- Si un système Trident indique qu'il se trouve dans le `failed` État bien qu'il ait auparavant travaillé, il est probable que cela soit causé par la modification des identifiants SVM/admin associés au back-end. Mise à jour des informations du back-end à l'aide de `tridentctl update backend` Vous pouvez également rebondir sur le pod Trident pour résoudre ce problème.
- Si vous mettez à niveau votre cluster Kubernetes et/ou Trident pour utiliser des snapshots de volume bêta, assurez-vous que tous les clichés alpha CRS existants sont entièrement supprimés. Vous pouvez ensuite utiliser le `tridentctl obliviate alpha-snapshot-crd` Commande permettant de supprimer des CRD alpha snapshot. Voir "de ce blog" pour comprendre les étapes de la migration des instantanés alpha.
- Si vous rencontrez des problèmes d'autorisation lors de l'installation de Trident avec Docker comme conteneur d'exécution, essayez d'installer Trident avec le `--in cluster=false` drapeau. Ceci n'utilise pas de module d'installation et évite les problèmes de permission observés en raison de l' `trident-installer` utilisateur.
- Utilisez le `uninstall parameter <Uninstalling Trident>` pour le nettoyage après un échec d'exécution. Par défaut, le script ne supprime pas les CRD créés par Trident, ce qui rend possible leur désinstallation et leur installation en toute sécurité, même dans le cadre d'un déploiement en cours d'exécution.
- Si vous souhaitez revenir à une version antérieure de Trident, exécutez d'abord le `tridentctl uninstall` Commande de suppression de Trident. Télécharger le fichier désiré "Version Trident" et installer à l'aide de `tridentctl install` commande. N'envisagez une mise à niveau vers une version antérieure qu'en l'absence de nouveaux volumes persistants créés et si aucune modification n'a été apportée aux classes de stockage PVS/back-end existantes. Trident utilise désormais des CRD pour la maintenance de l'état, toutes les entités de stockage créées (back-end, classes de stockage, volumes persistants et copies Snapshot de volumes) `associated CRD objects <Kubernetes CustomResourceDefinition Objects>` Au lieu d'écrire les données dans le volume persistant, qui a été utilisé par la version précédente de Trident installée. **Les nouveaux volumes persistants ne seront pas utilisables lors du déplacement vers une version antérieure. les modifications apportées aux objets, tels que les systèmes back-end, les volumes persistants, les classes de stockage et les instantanés de volumes (créés/mis à jour/supprimés) ne seront pas visibles dans Trident lors de la rétrogradation**. Le volume persistant utilisé par la version précédente de Trident installé reste visible par Trident. Si vous revenez à une version antérieure, l'accès aux volumes persistants n'est pas perturbé et ceux qui ont déjà été créés à l'aide de cette version antérieure, sauf s'ils ont été mis à niveau.
- Pour supprimer complètement Trident, exécutez le `tridentctl obliviate crd` commande. Ceci supprimera tous les objets CRD et dédéfinisse les CRD. Trident ne gérera plus les volumes persistants

déjà provisionnés.

> Trident devra être reconfiguré après coup.

- Après une installation réussie, si un PVC est bloqué dans le `Pending` phase, exécution `kubectl describe pvc` Peut fournir des informations supplémentaires sur les raisons pour lesquelles Trident n'a pas pu provisionner un volume persistant pour cette demande de volume persistant.

Dépannage d'un échec de déploiement Trident à l'aide de l'opérateur

Si vous déployez Trident à l'aide de l'opérateur, le statut de `TridentOrchestrator` modifications de `Installing` à `Installed`. Si vous observez l' `Failed` status, et l'opérateur ne peut pas récupérer en lui-même, il est recommandé de vérifier les journaux de l'opérateur en exécutant la commande suivante :

```
tridentctl logs -l trident-operator
```

Traînant les journaux du conteneur de l'opérateur trident peut pointer vers l'emplacement où se trouve le problème. Par exemple, un tel problème pourrait être l'impossibilité d'extraire les images de conteneur requises des registres en amont dans un environnement mis à l'air.

Pour comprendre pourquoi l'installation de Trident n'a pas été effectuée, consultez le `TridentOrchestrator` état.

```
kubectl describe torc trident-2
Name:         trident-2
Namespace:
Labels:       <none>
Annotations:  <none>
API Version:  trident.netapp.io/v1
Kind:         TridentOrchestrator
...
Status:
  Current Installation Params:
    IPv6:
    Autosupport Hostname:
    Autosupport Image:
    Autosupport Proxy:
    Autosupport Serial Number:
    Debug:
    Image Pull Secrets:         <nil>
    Image Registry:
    k8sTimeout:
    Kubelet Dir:
    Log Format:
    Silence Autosupport:
    Trident Image:
  Message:                      Trident is bound to another CR 'trident'
  Namespace:                    trident-2
  Status:                       Error
  Version:
Events:
  Type     Reason  Age                From                          Message
  ----     ------  ----               ----                          -------
  Warning  Error   16s (x2 over 16s)  trident-operator.netapp.io  Trident
is bound to another CR 'trident'
```

Cette erreur indique qu'il existe déjà un `TridentOrchestrator`Utilisé pour installer Trident. Étant donné que chaque cluster Kubernetes ne peut avoir qu'une seule instance de Trident, l'opérateur s'assure qu'une seule instance active existe à un instant donné `TridentOrchestrator` qu'il peut créer.

De plus, l'observation de l'état des pods Trident peut souvent indiquer si quelque chose n'est pas approprié.

```
kubectl get pods -n trident

NAME                                READY   STATUS             RESTARTS
AGE
trident-csi-4p5kq                   1/2     ImagePullBackOff   0
5m18s
trident-csi-6f45bfd8b6-vfrkw        4/5     ImagePullBackOff   0
5m19s
trident-csi-9q5xc                   1/2     ImagePullBackOff   0
5m18s
trident-csi-9v95z                   1/2     ImagePullBackOff   0
5m18s
trident-operator-766f7b8658-ldzsv   1/1     Running            0
8m17s
```

Vous pouvez clairement voir que les modules ne peuvent pas être initialisés complètement parce qu'une ou plusieurs images de conteneur n'ont pas été extraites.

Pour résoudre le problème, vous devez modifier le `TridentOrchestrator` CR. Vous pouvez également supprimer `TridentOrchestrator`, et en créer un nouveau avec la définition modifiée et précise.

Dépannage d'un déploiement Trident non réussi à l'aide de `tridentctl`

Pour vous aider à déterminer ce qui s'est mal passé, vous pouvez exécuter à nouveau le programme d'installation à l'aide du `-d` argument, qui active le mode débogage et vous aide à comprendre le problème :

```
./tridentctl install -n trident -d
```

Après avoir résolu le problème, vous pouvez nettoyer l'installation comme suit, puis exécuter le `tridentctl install` commande à nouveau :

```
./tridentctl uninstall -n trident
INFO Deleted Trident deployment.
INFO Deleted cluster role binding.
INFO Deleted cluster role.
INFO Deleted service account.
INFO Removed Trident user from security context constraint.
INFO Trident uninstallation succeeded.
```

Meilleures pratiques et recommandations

Déploiement

Utilisez les recommandations indiquées ici pour déployer Astra Trident.

Déploiement dans un namespace dédié

"Espaces de noms" séparation des tâches administratives entre les différentes applications et barrière au partage des ressources. Par exemple, un volume persistant ne peut pas être consommé depuis un autre espace de noms. Astra Trident fournit des ressources PV à tous les namespaces du cluster Kubernetes et exploite par conséquent un compte de service avec des privilèges élevés.

L'accès au pod Trident peut également permettre à un utilisateur d'accéder aux identifiants du système de stockage et à d'autres informations sensibles. Il est important de s'assurer que les utilisateurs d'applications et les applications de gestion ne peuvent pas accéder aux définitions d'objets Trident ou aux pods eux-mêmes.

Utilisez les quotas et les limites des plages pour contrôler la consommation du stockage

Kubernetes dispose de deux fonctionnalités qui, lorsqu'elles sont combinées, fournissent un mécanisme puissant pour limiter la consommation des ressources par les applications. Le "mécanisme de quotas de stockage" permet à l'administrateur d'implémenter des limites d'utilisation globales et spécifiques aux classes de stockage, à la capacité et au nombre d'objets, sur la base de chaque espace de noms. En outre, à l'aide d'un "limite de plage" Veille à ce que les demandes de volume persistant se situent dans une valeur minimale et maximale avant que la requête ne soit transférée au mécanisme de provisionnement.

Ces valeurs sont définies par espace de noms, ce qui signifie que chaque espace de noms doit avoir des valeurs définies qui correspondent à leurs besoins en ressources. Voir ici pour plus d'informations sur "comment exploiter les quotas".

Configuration de stockage sous-jacente

Chaque plateforme de stockage du portefeuille NetApp dispose de fonctionnalités uniques qui bénéficient aux applications, conteneurisées ou non.

Présentation de la plateforme

Trident fonctionne avec ONTAP et Element. Il n'existe pas de plate-forme mieux adaptée à toutes les applications et tous les scénarios qu'une autre. Cependant, les besoins de l'application et l'équipe chargée de l'administration du périphérique doivent être pris en compte lors du choix d'une plate-forme.

Vous devez suivre les meilleures pratiques de base du système d'exploitation hôte avec le protocole utilisé. Vous pouvez éventuellement envisager d'intégrer les meilleures pratiques des applications, le cas échéant, avec des paramètres de back-end, de classe de stockage et de volume persistant afin d'optimiser le stockage pour certaines applications.

Meilleures pratiques pour ONTAP et Cloud Volumes ONTAP

Découvrez les bonnes pratiques pour la configuration d'ONTAP et de Cloud Volumes ONTAP pour Trident.

Les recommandations suivantes sont des instructions de configuration de ONTAP pour les workloads conteneurisés, qui consomment des volumes provisionnés dynamiquement par Trident. Chaque élément doit être pris en compte et évalué en fonction de la pertinence dans votre environnement.

Utilisation de SVM(s) dédié(s) à Trident

Les machines virtuelles de stockage (SVM) assurent l'isolation et la séparation administrative entre les locataires sur un système ONTAP. La dédier un SVM aux applications permet de déléguer des privilèges et d'appliquer les meilleures pratiques en matière de limitation de la consommation des ressources.

Plusieurs options sont disponibles pour la gestion de la SVM :

- Fournir l'interface de gestion du cluster en configuration back-end avec les identifiants appropriés et spécifier le nom du SVM
- Créer une interface de gestion dédiée pour le SVM via ONTAP System Manager ou l'interface de ligne de commande.
- Partage du rôle de gestion avec une interface de données NFS

Dans chaque cas, l'interface doit être dans DNS et le nom DNS doit être utilisé lors de la configuration de Trident. Ainsi, certaines scénarios de reprise après incident, par exemple SVM-DR, sans conservation des identités de réseau.

Il n'existe aucune préférence entre une LIF de gestion dédiée ou partagée pour le SVM, cependant, vous devez vous assurer que vos stratégies de sécurité réseau sont en adéquation avec l'approche de votre choix. Quoi qu'il en soit, la LIF de gestion doit être accessible via DNS afin de faciliter une flexibilité maximale doit "SVM-DR" être utilisée conjointement avec Trident.

Limitez le nombre maximal de volumes

Les systèmes de stockage ONTAP disposent d'un nombre maximal de volumes, qui varie selon la version logicielle et la plateforme matérielle. Voir "NetApp Hardware Universe" Pour votre plateforme et votre version ONTAP afin de déterminer les limites exactes. Lorsque le nombre de volumes est épuisé, les opérations de provisionnement échouent non seulement pour Trident, mais pour l'ensemble des requêtes de stockage.

Trident `ontap-nas` et `ontap-san` Des pilotes provisionnent un volume flexible pour chaque volume persistant Kubernetes créé. Le `ontap-nas-economy` Le pilote crée environ un FlexVolume pour chaque 200 PVS (configurable entre 50 et 300). Le `ontap-san-economy` Le pilote crée environ un FlexVolume pour chaque 100 PVS (configurable entre 50 et 200). Pour empêcher Trident de consommer tous les volumes disponibles sur le système de stockage, vous devez définir une limite sur la SVM. Vous pouvez le faire à partir de la ligne de commande :

```
vserver modify -vserver <svm_name> -max-volumes <num_of_volumes>
```

La valeur pour `max-volumes` varie en fonction de plusieurs critères spécifiques à votre environnement :

- Le nombre de volumes existants dans le cluster ONTAP
- Le nombre de volumes que vous prévoyez de provisionner en dehors de Trident pour d'autres applications
- Nombre de volumes persistants que les applications Kubernetes devraient consommer

Le `max-volumes` Valeur est le volume total provisionné sur tous les nœuds du cluster ONTAP et non sur un nœud ONTAP individuel. Par conséquent, vous pouvez rencontrer des situations où un nœud de cluster

ONTAP peut avoir plus ou moins de volumes provisionnés Trident qu'un autre nœud.

Par exemple, un cluster ONTAP à deux nœuds peut héberger un maximum de 2000 volumes flexibles. Avoir le volume maximum réglé sur 1250 semble très raisonnable. Cependant, si seulement "64 bits" Depuis un nœud est attribué à la SVM, ou les agrégats attribués à partir d'un nœud ne peuvent pas être provisionnés sur (par exemple, en raison de la capacité). L'autre nœud devient alors la cible de tous les volumes provisionnés par Trident. Cela signifie que le volume peut être atteint en limite pour ce nœud avant le `max-volumes` La valeur est atteinte, ce qui affecte Trident et les autres opérations de volume utilisant ce nœud. **Vous pouvez éviter cette situation en vous assurant que les agrégats de chaque nœud du cluster sont attribués à la SVM utilisée par Trident en chiffres égaux.**

Limitez la taille maximale des volumes créés par Trident

Pour configurer la taille maximale des volumes pouvant être créés par Trident, utilisez la `limitVolumeSize` dans votre `backend.json` définition.

Vous devez aussi exploiter les fonctionnalités Kubernetes pour contrôler la taille du volume au niveau de la baie de stockage.

Configurez Trident pour utiliser le protocole CHAP bidirectionnel

Vous pouvez spécifier l'initiateur CHAP et les noms d'utilisateur et mots de passe cibles dans votre définition du système back-end et activer Trident sur la SVM. À l'aide du `useCHAP` Paramètre dans votre configuration back-end, Trident authentifie les connexions iSCSI pour les systèmes back-end ONTAP avec CHAP. La prise en charge CHAP bidirectionnelle est disponible avec Trident 20.04 et versions ultérieures.

Création et utilisation d'une politique de QoS de SVM

L'utilisation d'une politique de QoS de ONTAP appliquée au SVM limite le nombre de consommables d'IOPS par les volumes provisionnés par Trident. Cela permet à "évitez un tyran" ou à un conteneur hors contrôle d'affecter les charges de travail en dehors de la SVM Trident.

Vous pouvez créer une politique de QoS pour la SVM en quelques étapes. Consultez la documentation de votre version de ONTAP pour obtenir des informations précises. L'exemple ci-dessous crée une politique de QoS qui limite le nombre total d'IOPS disponibles pour la SVM à 5000.

```
# create the policy group for the SVM
qos policy-group create -policy-group <policy_name> -vserver <svm_name>
-max-throughput 5000iops

# assign the policy group to the SVM, note this will not work
# if volumes or files in the SVM have existing QoS policies
vserver modify -vserver <svm_name> -qos-policy-group <policy_name>
```

Si votre version d'ONTAP la prend en charge, il est également possible d'utiliser un minimum de QoS pour garantir un débit minimum pour les workloads conteneurisés. La QoS adaptative n'est pas compatible avec une règle de niveau SVM.

Le nombre d'IOPS dédiées aux workloads conteneurisés dépend de plusieurs aspects. Entre autres choses :

- Autres charges de travail qui utilisent la baie de stockage. Si certaines charges de travail, autres que celles liées au déploiement Kubernetes avec les ressources de stockage, veillez à ne pas affecter

accidentellement ces charges de travail.

- Workloads attendus s'exécutant dans des conteneurs. Si des charges de travail qui exigent des IOPS élevées s'exécutent dans des conteneurs, une faible politique de QoS entraîne une mauvaise expérience.

Il est important de rappeler qu'une politique de QoS attribuée au niveau du SVM entraîne tous les volumes provisionnés sur la SVM et partageant le même pool d'IOPS. Si l'une des applications conteneurisées a une exigence d'IOPS élevées, elle pourrait devenir une force dominante pour les autres workloads conteneurisés. Dans ce cas, vous pourriez envisager d'utiliser l'automatisation externe pour attribuer des règles de QoS par volume.

Vous devez affecter la « policy group » QoS à la SVM **Only** si la version de votre ONTAP est antérieure à 9.8.

Création de groupes de règles de QoS pour Trident

La qualité de service (QoS) garantit que les performances des workloads stratégiques ne sont pas dégradées par des charges de travail concurrentes. Les groupes de règles de QoS de ONTAP proposent des options de QoS pour les volumes et permettent aux utilisateurs de définir le plafond de débit pour une ou plusieurs charges de travail. Pour plus d'informations sur QoS, voir "Débit garanti avec la QoS". Vous pouvez spécifier des groupes de règles de QoS dans le back-end ou dans un pool de stockage, et ils sont appliqués à chaque volume créé dans ce pool ou back-end.

ONTAP propose deux types de groupes de règles de QoS : classiques et évolutifs. Les groupes de règles classiques fournissent un débit minimal (ou minimal, dans les versions ultérieures) plat en IOPS. La QoS adaptative ajuste automatiquement le débit en fonction de la taille du workload. Elle maintient le rapport entre les IOPS et les To|Go en fonction de l'évolution de la taille du workload. Vous pouvez ainsi gérer des centaines, voire des milliers de charges de travail dans le cadre d'un déploiement à grande échelle.

Avant de créer des groupes de règles de QoS, tenez compte des points suivants :

- Vous devez définir le `qosPolicy` saisissez le `defaults` bloc de la configuration back-end. Voir l'exemple de configuration back-end suivant :

```
  ---
version: 1
storageDriverName: ontap-nas
managementLIF: 0.0.0.0
dataLIF: 0.0.0.0
svm: svm0
username: user
password: pass
defaults:
  qosPolicy: standard-pg
storage:
- labels:
    performance: extreme
  defaults:
    adaptiveQosPolicy: extremely-adaptive-pg
- labels:
    performance: premium
  defaults:
    qosPolicy: premium-pg
```

- Vous devez appliquer les « policy groups » par volume pour que chaque volume bénéficie de l'intégralité du débit spécifié par le « policy group ». Les groupes de stratégies partagés ne sont pas pris en charge.

Pour plus d'informations sur les groupes de règles QoS, reportez-vous àla section "Commandes QoS de ONTAP 9.8".

Limitez l'accès aux ressources de stockage aux membres du cluster Kubernetes

La limitation de l'accès aux volumes NFS et aux LUN iSCSI créés par Trident est un composant stratégique du niveau de sécurité pour votre déploiement Kubernetes. En effet, les hôtes qui ne font pas partie du cluster Kubernetes n'accèdent pas aux volumes et peuvent modifier les données de façon inattendue.

Il est important de comprendre que les espaces de noms sont la limite logique des ressources dans Kubernetes. L'hypothèse est que les ressources dans un même espace de noms peuvent être partagées, mais, surtout, il n'existe aucune fonctionnalité de multi-espace de noms. Même si les volumes persistants sont des objets globaux, lorsqu'ils sont liés à une demande de volume persistant, ils ne sont accessibles que par des pods qui se trouvent dans le même espace de noms. **Il est essentiel de s'assurer que les espaces de noms sont utilisés pour fournir la séparation, le cas échéant.**

La préoccupation principale de la plupart des entreprises en ce qui concerne la sécurité des données dans un contexte Kubernetes est qu'un processus dans un conteneur peut accéder au stockage monté sur l'hôte, mais qui n'est pas destiné au conteneur. "Espaces de noms" sont conçus pour éviter ce type de compromis. Toutefois, il y a une exception : les conteneurs privilégiés.

Un conteneur privilégié est un conteneur exécuté avec beaucoup plus d'autorisations au niveau de l'hôte que la normale. Par défaut, ces dernières ne sont pas refusées. Veillez donc à désactiver cette fonctionnalité en utilisant "stratégies de sécurité des pods".

Pour les volumes pour lesquels l'accès est demandé depuis Kubernetes et des hôtes externes, le stockage doit être géré de manière classique, avec le volume persistant introduit par l'administrateur et non géré par

Trident. Cela garantit que le volume de stockage est détruit uniquement lorsque les hôtes Kubernetes et externes sont déconnectés et qu'ils n'utilisent plus le volume. En outre, il est possible d'appliquer une export policy personnalisée qui permet l'accès depuis les nœuds de cluster Kubernetes et les serveurs ciblés à l'extérieur du cluster Kubernetes.

Pour les déploiements avec des nœuds d'infrastructure dédiés (par exemple OpenShift) ou d'autres nœuds ne pouvant pas planifier les applications utilisateur, des règles d'exportation distinctes doivent être utilisées pour limiter davantage l'accès aux ressources de stockage. Cela inclut la création d'une export policy pour les services qui sont déployés sur ces nœuds d'infrastructure (par exemple les services OpenShift Metrics et Logging Services), ainsi que pour les applications standard déployées sur des nœuds non liés à l'infrastructure.

Utiliser une export policy dédiée

Vous devez vous assurer qu'il existe une export policy pour chaque backend qui autorise uniquement l'accès aux nœuds présents dans le cluster Kubernetes. Trident peut créer et gérer automatiquement des règles d'exportation depuis la version 20.04. Trident limite ainsi l'accès aux volumes qu'il provisionne aux nœuds du cluster Kubernetes et simplifie l'ajout et la suppression des nœuds.

Vous pouvez également créer une export policy manuellement et la remplir à l'aide d'une ou plusieurs règles d'exportation qui traitent chaque demande d'accès de nœud :

- Utilisez le `vserver export-policy create` Commande CLI ONTAP pour créer l'export policy.
- Ajoutez des règles à la export policy à l'aide de `vserver export-policy rule create` Commande CLI ONTAP.

L'exécution de ces commandes vous permet de limiter l'accès aux données aux nœuds Kubernetes.

Désactiver `showmount` Pour le SVM applicatif

Le `showmount` Cette fonctionnalité permet à un client NFS d'interroger le SVM pour obtenir la liste des exportations NFS disponibles. Un pod déployé sur le cluster Kubernetes peut lancer le `showmount -e` Commande au niveau de la LIF de données et reçoit la liste des montages disponibles, y compris ceux auxquels elle n'a pas accès. Bien qu'il ne s'agisse pas d'un compromis sur la sécurité, cette solution fournit des informations inutiles susceptibles d'aider un utilisateur non autorisé à se connecter à une exportation NFS.

Vous devez désactiver `showmount` En utilisant la commande CLI ONTAP au niveau du SVM :

```
vserver nfs modify -vserver <svm_name> -showmount disabled
```

Les meilleures pratiques pour SolidFire

Découvrez les bonnes pratiques pour la configuration du stockage SolidFire pour Trident.

Créer un compte SolidFire

Chaque compte SolidFire représente un propriétaire de volume unique et reçoit ses propres informations d'identification CHAP (Challenge-Handshake Authentication Protocol). Vous pouvez accéder aux volumes affectés à un compte en utilisant le nom du compte et les informations d'identification CHAP relatives ou par le biais d'un groupe d'accès de volume. Un compte peut comporter jusqu'à deux milliers de volumes qui lui sont attribués, mais un volume ne peut appartenir qu'à un seul compte.

Création d'une règle de QoS

Utilisez les règles de QoS SolidFire pour créer et enregistrer des paramètres de qualité de service standardisés qui peuvent être appliqués à de nombreux volumes.

Vous pouvez définir des paramètres de QoS par volume. Les performances de chaque volume peuvent être garanties en définissant trois paramètres configurables pour définir les QoS : IOPS min, IOPS max et IOPS en rafale.

Voici les valeurs d'IOPS minimales, maximales et en rafale possibles pour la taille de bloc de 4 Ko.

Paramètre IOPS	Définition	Minimum valeur	Valeur par défaut	Capacité Valeur (4 Ko)
IOPS min	Niveau de performance garanti pour un volume.	50	50	15000
IOPS max	La performance ne dépassera pas cette limite.	50	15000	200,000
IOPS en rafale	IOPS maximales autorisées en rafale,	50	15000	200,000

Même si les IOPS maximales et en rafale peuvent être définies jusqu'à 200,000, les performances maximales réelles d'un volume sont limitées par l'utilisation du cluster et les performances par nœud.

La taille et la bande passante des blocs influencent directement le nombre d'opérations d'entrée/sortie par seconde. Lorsque la taille de bloc augmente, le système augmente la bande passante jusqu'au niveau nécessaire pour traiter les tailles de bloc de taille supérieure. Lorsque la bande passante augmente, le nombre d'IOPS augmente, le système peut atteindre une baisse. Voir "Qualité de service SolidFire" Pour plus d'informations sur la qualité de service et les performances.

Authentification SolidFire

Element prend en charge deux méthodes d'authentification : CHAP et VAG (Volume Access Groups). CHAP utilise le protocole CHAP pour authentifier l'hôte au back-end. Les groupes d'accès de volume contrôlent l'accès aux volumes qu'ils provisionne. NetApp recommande d'utiliser le protocole CHAP pour l'authentification, car il est plus simple et ne comporte pas de limites d'évolutivité.

Trident avec le mécanisme de provisionnement CSI amélioré prend en charge l'authentification CHAP. Les VAGs ne doivent être utilisés que dans le mode de fonctionnement traditionnel non CSI.

L'authentification CHAP (vérification que l'initiateur est l'utilisateur de volume prévu) n'est prise en charge qu'avec un contrôle d'accès basé sur le compte. Si vous utilisez CHAP pour l'authentification, deux options sont disponibles : CHAP unidirectionnel et CHAP bidirectionnel. L'authentification CHAP unidirectionnelle authentifie l'accès au volume à l'aide du nom du compte SolidFire et du secret de l'initiateur. L'option CHAP bidirectionnelle fournit le moyen le plus sûr d'authentifier le volume car le volume authentifie l'hôte via le nom du compte et le secret de l'initiateur, puis l'hôte authentifie le volume via le nom du compte et le secret cible.

Toutefois, si CHAP ne peut pas être activé et que VAGs sont requis, créez le groupe d'accès et ajoutez les initiateurs hôtes et les volumes au groupe d'accès. Chaque IQN que vous ajoutez à un groupe d'accès peut accéder à chaque volume du groupe avec ou sans authentification CHAP. Si l'initiateur iSCSI est configuré pour utiliser l'authentification CHAP, un contrôle d'accès basé sur les comptes est utilisé. Si l'initiateur iSCSI n'est pas configuré pour utiliser l'authentification CHAP, le contrôle d'accès au groupe d'accès de volume est utilisé.

Où trouver plus d'informations ?

Une partie de la documentation sur les meilleures pratiques est présentée ci-dessous. Rechercher dans le "Bibliothèque NetApp" pour les versions les plus récentes.

ONTAP

- "Guide des meilleures pratiques et de mise en œuvre de NFS"
- "Gestion du stockage SAN" (Pour iSCSI)
- "Configuration iSCSI Express pour RHEL"

Logiciel Element

- "Configuration de SolidFire pour Linux"

NetApp HCI

- "Conditions préalables au déploiement de NetApp HCI"
- "Accès au moteur de déploiement NetApp"

Information sur les pratiques exemplaires des applications

- "Bonnes pratiques pour MySQL sur ONTAP"
- "Bonnes pratiques pour MySQL sur SolidFire"
- "NetApp SolidFire et Cassandra"
- "Meilleures pratiques pour Oracle sur SolidFire"
- "Meilleures pratiques PostgreSQL sur SolidFire"

Toutes les applications ne disposent pas d'instructions spécifiques, il est important de collaborer avec votre équipe NetApp et d'utiliser le "Bibliothèque NetApp" pour trouver la documentation la plus récente.

Intégrez Astra Trident

Pour intégrer Astra Trident, les éléments de conception et d'architecture suivants nécessitent l'intégration : sélection des pilotes et déploiement, conception de la classe de stockage, conception de pool virtuel, impact de la demande de volume persistant sur le provisionnement du stockage, les opérations des volumes et le déploiement de services OpenShift avec Astra Trident.

Choix et déploiement du conducteur

Sélectionnez et déployez un pilote backend pour votre système de stockage.

Pilotes ONTAP backend

Les pilotes back-end ONTAP sont différenciés par le protocole utilisé et le mode de provisionnement des volumes sur le système de stockage. Par conséquent, réfléchissez bien au choix du conducteur à déployer.

À un niveau plus élevé, si votre application dispose de composants qui nécessitent un stockage partagé (plusieurs modules accédant au même volume de demande de volume persistant), les pilotes NAS seraient la solution par défaut, tandis que les pilotes iSCSI basés sur les blocs répondent aux besoins d'un stockage non partagé. Choisir le protocole en fonction des besoins de l'application et du niveau de confort des équipes chargées du stockage et de l'infrastructure. En règle générale, ces différences sont peu nombreuses pour la plupart des applications. La décision dépend donc souvent de la nécessité d'un stockage partagé (dans lequel plusieurs pods auront besoin d'un accès simultané).

Les pilotes ONTAP backend disponibles sont les suivants :

- `ontap-nas`: Chaque volume persistant provisionné est un volume flexible ONTAP complet.
- `ontap-nas-economy`: Chaque volume persistant provisionné est un qtree, avec un nombre configurable de qtrees par FlexVolume (la valeur par défaut est 200).
- `ontap-nas-flexgroup`: Chaque volume persistant provisionné en tant que ONTAP FlexGroup complet et tous les agrégats affectés à un SVM sont utilisés.
- `ontap-san`: Chaque volume persistant provisionné est un LUN au sein de son propre volume FlexVolume.
- `ontap-san-economy`: Chaque volume persistant provisionné est une LUN, avec un nombre configurable de LUN par FlexVolume (la valeur par défaut est 100).

Le choix entre les trois pilotes NAS a des ramifications sur les fonctionnalités mises à disposition de l'application.

Il est à noter que dans les tableaux ci-dessous, toutes les fonctionnalités ne sont pas exposées par Astra Trident. L'administrateur du stockage doit appliquer une partie après le provisionnement si cette fonctionnalité est souhaitée. Les notes de bas de page en exposant distinguent les fonctionnalités par fonction et pilote.

Pilotes NAS ONTAP	Snapshots	Clones	Règles d'exportation dynamiques	Multi-attacher	La QoS	Redimensionner	La réplication
`ontap-nas`	Oui.	Oui.	Yes [5]	Oui.	Yes [1]	Oui.	Yes [1]
`ontap-nas-economy`	Yes [3]	Yes [3]	Yes [5]	Oui.	Yes [3]	Oui.	Yes [3]
`ontap-nas-flexgroup`	Yes [1]	Non	Yes [5]	Oui.	Yes [1]	Oui.	Yes [1]

Astra Trident propose 2 pilotes SAN pour ONTAP dont les fonctionnalités sont présentées ci-dessous.

Pilotes SAN de ONTAP	Snapshots	Clones	Multi-attacher	Chap bi-directionnel	La QoS	Redimensionner	La réplication
`ontap-san`	Oui.	Oui.	Yes [4]	Oui.	Yes [1]	Oui.	Yes [1]

Pilotes SAN de ONTAP	Snapshots	Clones	Multi-attacher	Chap bi-directionnel	La QoS	Redimensionner	La réplication
`ontap-san-economy`	Oui.	Oui.	Yes [4]	Oui.	Yes [3]	Oui.	Yes [3]

> Note de bas de page pour les tableaux ci-dessus : Yes [1]: Non géré par Astra Trident Yes [2]: Géré par Astra Trident, mais pas PV granulaire Yes [3]: Non géré par Astra Trident et non PV granulaire Yes [4]: Supporté par 5 Trident pour les volumes en mode bloc brut[]: Supporté par Trident

Les fonctionnalités qui ne sont pas granulaires volume persistant sont appliquées à l'ensemble du volume flexible et tous les volumes persistants (qtrees ou LUN inclus dans les volumes FlexVol partagés) partageront une planification commune.

Comme on peut le voir dans les tableaux ci-dessus, une grande partie des fonctionnalités entre `ontap-nas` et `ontap-nas-economy` est identique. Cependant, parce que le `ontap-nas-economy` Le pilote limite la capacité à contrôler la planification à la granularité par volume persistant, ce qui peut affecter en particulier la reprise après incident et la planification des sauvegardes. Pour les équipes de développement qui souhaitent exploiter la fonctionnalité de clonage PVC sur le stockage ONTAP, ce n'est possible que lorsque vous utilisez le `ontap-nas`, `ontap-san` ou `ontap-san-economy` pilotes.

> Le `solidfire-san` Le pilote est également capable de cloner des demandes de volume persistant.

Pilotes Cloud Volumes ONTAP backend

Cloud Volumes ONTAP assure le contrôle des données et des fonctionnalités de stockage haute performance dans divers cas d'utilisation, notamment pour les partages de fichiers et le stockage de niveau bloc qui servent les protocoles NAS et SAN (NFS, SMB/CIFS et iSCSI). Les pilotes compatibles avec Cloud Volume ONTAP sont les `ontap-nas`, `ontap-nas-economy`, `ontap-san` et `ontap-san-economy`. Applicable à Cloud Volume ONTAP pour Azure, Cloud Volume ONTAP pour GCP.

Pilotes backend Amazon FSX pour ONTAP

Avec Amazon FSX pour ONTAP, les clients peuvent exploiter les fonctions, les performances et les fonctionnalités d'administration NetApp qu'ils connaissent bien, tout en bénéficiant de la simplicité, de l'agilité, de la sécurité et de l'évolutivité du stockage de données sur AWS. FSX pour ONTAP prend en charge de nombreuses fonctionnalités de système de fichiers et API d'administration d'ONTAP. Les pilotes compatibles avec Cloud Volume ONTAP sont les `ontap-nas`, `ontap-nas-economy`, `ontap-nas-flexgroup`, `ontap-san` et `ontap-san-economy`.

Pilotes back-end NetApp HCI/SolidFire

Le `solidfire-san` Pilote utilisé avec les plateformes NetApp HCI/SolidFire pour aider l'administrateur à configurer un back-end Element pour Trident sur la base des limites de QoS. Si vous voulez concevoir votre système back-end pour définir les limites de QoS spécifiques sur les volumes provisionnés par Trident, utilisez la `type` paramètre dans le fichier backend. L'administrateur peut également restreindre la taille du volume pouvant être créé sur le stockage à l'aide de `limitVolumeSize` paramètre. Pour le moment, les fonctionnalités de stockage Element telles que le redimensionnement des volumes et la réplication des volumes ne sont pas prises en charge via le `solidfire-san` conducteur. Ces opérations doivent être effectuées manuellement via l'interface utilisateur Web du logiciel Element.

Pilote SolidFire	Snapshots	Clones	Multi-attacher	CHAP	La QoS	Redimensionner	La réplication
`solidfire-san`	Oui.	Oui.	Yes [2]	Oui.	Oui.	Oui.	Yes [1]

> Note de bas de page: Yes [1]: Non géré par Astra Trident Yes [2]: Pris en charge pour les volumes en blocs bruts

Pilotes Azure NetApp Files backend

Astra Trident utilise le `azure-netapp-files` pilote pour gérer le "Azure NetApp Files" services.

Vous trouverez plus d'informations sur ce pilote et sa configuration dans le "Configuration back-end d'Astra Trident pour Azure NetApp Files".

Pilote Azure NetApp Files	Snapshots	Clones	Multi-attacher	La QoS	Développement	La réplication
`azure-netapp-files`	Oui.	Oui.	Oui.	Oui.	Oui.	Yes [1]

> Note de bas de page: Yes [1]: Non géré par Astra Trident

Cloud Volumes Service sur le pilote back-end Google Cloud

Astra Trident utilise le `gcp-cvs` Pilote de liaison avec Cloud Volumes Service sur Google Cloud.

Le `gcp-cvs` Le pilote utilise des pools virtuels pour extraire le système back-end et permettre à Astra Trident de déterminer le placement des volumes. L'administrateur définit les pools virtuels dans `backend.json` fichiers. Les classes de stockage utilisent des sélecteurs pour identifier les pools virtuels par étiquette.

- Si des pools virtuels sont définis au niveau du système back-end, Astra Trident essaie de créer un volume dans les pools de stockage Google Cloud auxquels ces pools virtuels sont limités.
- Si des pools virtuels ne sont pas définis dans le système back-end, Astra Trident sélectionne un pool de stockage Google Cloud à partir des pools de stockage disponibles dans la région.

Pour configurer le back-end Google Cloud avec Astra Trident, vous devez préciser `projectNumber`, `apiRegion`, et `apiKey` dans le fichier backend. Le numéro de projet est indiqué dans la console Google Cloud. La clé API est utilisée depuis le fichier de clé privée du compte de service que vous avez créé lors de la configuration de l'accès API pour Cloud Volumes Service sur Google Cloud.

Pour plus d'informations sur les types de services et les niveaux de service de Cloud Volumes Service sur Google Cloud, consultez la page "En savoir plus sur la prise en charge d'Astra Trident pour CVS pour GCP".

Pilote Cloud Volumes Service pour Google Cloud	Snapshots	Clones	Multi-attacher	La QoS	Développement	La réplication
`gcp-cvs`	Oui.	Oui.	Oui.	Oui.	Oui.	Disponible uniquement sur le type de service CVS-Performance.

Notes de réplication

- La réplication n'est pas gérée par Astra Trident.
- Le clone sera créé dans le même pool de stockage que le volume source.

Conception de classe de stockage

Chaque classe de stockage doit être configurée et appliquée pour créer un objet de classe de stockage Kubernetes. Cette section décrit comment concevoir un système de stockage pour votre application.

Utilisation du système back-end spécifique

Le filtrage peut être utilisé au sein d'un objet de classe de stockage spécifique pour déterminer le pool de stockage ou l'ensemble de pools à utiliser avec cette classe de stockage spécifique. Trois ensembles de filtres peuvent être définis dans la classe de stockage : `storagePools`, `additionalStoragePools`, et/ou `excludeStoragePools`.

Le `storagePools` paramètre permet de limiter le stockage à l'ensemble de pools correspondant à tous les attributs spécifiés. Le `additionalStoragePools` Le paramètre est utilisé pour étendre l'ensemble de pools qu'Astra Trident utilisera pour le provisionnement ainsi que l'ensemble de pools sélectionnés par les attributs et `storagePools` paramètres. Vous pouvez utiliser l'un ou l'autre paramètre seul ou les deux ensemble pour vous assurer que l'ensemble approprié de pools de stockage est sélectionné.

Le `excludeStoragePools` le paramètre est utilisé pour exclure spécifiquement l'ensemble de pools répertoriés qui correspondent aux attributs.

Émuler les règles de QoS

Si vous souhaitez concevoir des classes de stockage pour émuler les règles de qualité de service, créez une classe de stockage avec le `media` attribut en tant que `hdd` ou `ssd`. Basé sur `media` Attribut mentionné dans la classe de stockage, Trident sélectionne le back-end approprié qui sert `hdd` ou `ssd` les agrégats correspondent à l'attribut du support, puis dirigent le provisionnement des volumes sur l'agrégat spécifique. Nous pouvons donc créer une PRIME de classe de stockage qui aurait été nécessaire `media` attribut défini comme `ssd` Qui peuvent être classées comme politique DE qualité de service PREMIUM. Nous pouvons créer une autre NORME de classe de stockage dont l'attribut de support est défini comme `hdd', qui pourrait être classé comme règle de QoS STANDARD. Nous pourrions également utiliser l'attribut « IOPS » de la classe de stockage pour rediriger le provisionnement vers une appliance Element qui peut être définie comme une règle de QoS.

Utilisation du système back-end en fonction de fonctionnalités spécifiques

Les classes de stockage peuvent être conçues pour diriger le provisionnement des volumes sur un système back-end spécifique, où des fonctionnalités telles que le provisionnement fin et lourd, les copies Snapshot, les clones et le chiffrement sont activées. Pour spécifier le stockage à utiliser, créez des classes de stockage qui spécifient le back-end approprié avec la fonction requise activée.

Pools virtuels

Des pools virtuels sont disponibles pour tous les systèmes back-end Astra Trident. Vous pouvez définir des pools virtuels pour tout système back-end, à l'aide de n'importe quel pilote fourni par Astra Trident.

Les pools virtuels permettent à un administrateur de créer un niveau d'abstraction sur les systèmes back-end, qui peut être référencé via des classes de stockage, pour une plus grande flexibilité et un placement efficace des volumes dans les systèmes back-end. Différents systèmes back-end peuvent être définis avec la même classe de service. En outre, il est possible de créer plusieurs pools de stockage sur le même back-end, mais avec des caractéristiques différentes. Lorsqu'une classe de stockage est configurée avec un sélecteur portant les étiquettes spécifiques, Astra Trident choisit un système back-end correspondant à toutes les étiquettes de sélection pour placer le volume. Si les étiquettes de sélection de classe de stockage correspondent à plusieurs pools de stockage, Astra Trident choisira l'un d'entre eux pour provisionner le volume.

Conception de pool virtuel

Lors de la création d'un backend, vous pouvez généralement spécifier un ensemble de paramètres. Il était impossible pour l'administrateur de créer un autre système back-end avec les mêmes identifiants de stockage et avec un ensemble de paramètres différent. Grâce à l'introduction de pools virtuels, ce problème a été résolu. Les pools virtuels sont une abstraction de niveau introduite entre le back-end et la classe de stockage Kubernetes. L'administrateur peut ainsi définir des paramètres et des étiquettes que l'on peut référencer via les classes de stockage Kubernetes comme un sélecteur, de façon indépendante du back-end. Il est possible de définir des pools virtuels pour tous les systèmes back-end NetApp pris en charge avec Astra Trident. Il s'agit notamment des systèmes SolidFire/NetApp HCI, ONTAP, Cloud Volumes Service sur GCP et Azure NetApp Files.

> Lors de la définition de pools virtuels, il est recommandé de ne pas tenter de réorganiser l'ordre des pools virtuels existants dans une définition backend. Il est également conseillé de ne pas modifier/modifier les attributs d'un pool virtuel existant et de définir un nouveau pool virtuel à la place.

Émulation de différents niveaux de service/QoS

Il est possible de concevoir des pools virtuels pour émuler des classes de service. Grâce à l'implémentation du pool virtuel pour Cloud volumes Service pour Azure NetApp Files, examinons comment nous pouvons configurer différentes classes de service. Configurez le back-end ANF avec plusieurs étiquettes représentant différents niveaux de performance. Réglez `servicelevel` aspect au niveau de performance approprié et ajouter d'autres aspects requis sous chaque étiquette. Créez désormais différentes classes de stockage Kubernetes qui seraient mappées sur différents pools virtuels. À l'aide du `parameters.selector` Chaque classe de stockage indique quels pools virtuels peuvent être utilisés pour héberger un volume.

Attribution d'un ensemble spécifique d'aspects

Il est possible de concevoir plusieurs pools virtuels, dont les aspects sont spécifiques, à partir d'un système back-end unique. Pour ce faire, configurez le back-end avec plusieurs étiquettes et définissez les aspects requis sous chaque étiquette. Créez désormais des classes de stockage Kubernetes différentes avec le `parameters.selector` champ correspondant à différents pools virtuels. Les volumes provisionnés sur le

back-end possèdent les aspects définis dans le pool virtuel choisi.

Caractéristiques des PVC qui affectent le provisionnement du stockage

Certains paramètres au-delà de la classe de stockage requise peuvent affecter le processus de décision de provisionnement Astra Trident lors de la création d'un volume persistant.

Mode d'accès

Lors de la demande de stockage via un PVC, l'un des champs obligatoires est le mode d'accès. Le mode désiré peut affecter le back-end sélectionné pour héberger la demande de stockage.

Astra Trident tentera de correspondre au protocole de stockage utilisé avec la méthode d'accès spécifiée dans la matrice suivante. Cette technologie est indépendante de la plateforme de stockage sous-jacente.

	ReadWriteOnce	ReadOnlyMany	ReadWriteMany
ISCSI	Oui.	Oui.	Oui (bloc brut)
NFS	Oui.	Oui.	Oui.

Toute demande de volume persistant ReadWriteMany soumise à un déploiement Trident sans système back-end NFS configuré entraînera le provisionnement d'un volume. Pour cette raison, le demandeur doit utiliser le mode d'accès qui convient à son application.

Opérations de volume

Modifier les volumes persistants

Les volumes persistants sont, à deux exceptions près, des objets immuables dans Kubernetes. Une fois créée, la règle de récupération et la taille peuvent être modifiées. Toutefois, cela n'empêche pas la modification de certains aspects du volume en dehors de Kubernetes. Vous pouvez ainsi personnaliser le volume pour des applications spécifiques, en veillant à ce que la capacité ne soit pas accidentellement consommée ou tout simplement pour déplacer le volume vers un autre contrôleur de stockage pour n'importe quelle raison.

Les actuellement sur provisionnement des arborescences Kubernetes ne prennent pas en charge les opérations de redimensionnement des volumes pour les volumes NFS ou iSCSI PVS. Astra Trident prend en charge l'extension des volumes NFS et iSCSI.

Les détails de connexion du PV ne peuvent pas être modifiés après sa création.

Création de copies Snapshot de volume à la demande

Astra Trident prend en charge la création de copies Snapshot de volume à la demande et la création de demandes de volume persistant à partir de copies Snapshot via le framework CSI. Les snapshots constituent une méthode pratique de conservation des copies ponctuelles des données et ont un cycle de vie indépendant du volume persistant source dans Kubernetes. Ces snapshots peuvent être utilisés pour cloner des demandes de volume persistant.

Créer des volumes à partir de copies Snapshot

Astra Trident prend également en charge la création de volumes persistant à partir des snapshots de volume. Pour ce faire, il suffit de créer une demande de volume persistant et de mentionner le `datasource` l'instantané requis à partir duquel le volume doit être créé. Astra Trident va gérer ce volume de volume

persistant en créant un volume dont les données sont présentes sur le snapshot. Grâce à cette fonctionnalité, il est possible de dupliquer des données entre régions, de créer des environnements de test, de remplacer un volume de production endommagé ou corrompu dans son intégralité, ou de récupérer des fichiers et des répertoires spécifiques et de les transférer vers un autre volume attaché.

Déplacement des volumes dans le cluster

Les administrateurs du stockage peuvent déplacer des volumes entre les agrégats et les contrôleurs du cluster ONTAP sans interruption pour l'utilisateur du stockage. Cette opération n'affecte pas Astra Trident ou le cluster Kubernetes, tant que l'agrégat de destination est un auquel le SVM utilisé par Astra Trident a accès. Important : si l'agrégat a été récemment ajouté au SVM, le système back-end devra être actualisé en le ajoutant à Astra Trident. Cela déclenchera l'Astra Trident afin de réinventorier la SVM afin que le nouvel agrégat soit reconnu.

Néanmoins, Astra Trident ne prend pas automatiquement en charge le déplacement des volumes entre les systèmes back-end. Il s'agit notamment d'étendre les SVM au sein d'un même cluster, entre plusieurs clusters ou sur une autre plateforme de stockage (même si ce système est un SVM connecté à Astra Trident).

Si un volume est copié à un autre emplacement, la fonctionnalité d'importation de volume peut être utilisée pour importer les volumes actuels dans Astra Trident.

Développement des volumes

Astra Trident prend en charge le redimensionnement des volumes NFS et iSCSI PVS. Les utilisateurs peuvent ainsi redimensionner leurs volumes directement via la couche Kubernetes. L'extension de volume est possible pour toutes les principales plateformes de stockage NetApp, y compris ONTAP, SolidFire/NetApp HCI et les systèmes back-end Cloud Volumes Service. Pour permettre une extension possible ultérieurement, définissez `allowVolumeExpansion` à `true` Dans votre classe de stockage associée au volume. Lorsque le volume persistant doit être redimensionné, modifiez le `spec.resources.requests.storage` Annotation dans la demande de volume persistant vers la taille de volume requise. Trident s'occupe automatiquement du redimensionnement du volume sur le cluster de stockage.

Importer un volume existant dans Kubernetes

L'importation de volumes permet d'importer un volume de stockage existant dans un environnement Kubernetes. Cette opération est actuellement prise en charge par `ontap-nas`, `ontap-nas-flexgroup`, `solidfire-san`, `azure-netapp-files`, et `gcp-cvs` pilotes. Cette fonctionnalité est utile lors du portage d'une application existante sur Kubernetes ou lors de scénarios de reprise après incident.

Lorsque vous utilisez ONTAP et `solidfire-san` pilotes, utilisez la commande `tridentctl import volume <backend-name> <volume-name> -f /path/pvc.yaml` Pour importer un volume existant dans Kubernetes et le gérer par Astra Trident. Le fichier PVC YAML ou JSON utilisé dans la commande de volume d'importation pointe vers une classe de stockage qui identifie Astra Trident comme provisionneur. Si vous utilisez un système back-end NetApp HCI/SolidFire, assurez-vous que les noms des volumes sont uniques. Si les noms des volumes sont dupliqués, cloner le volume en un nom unique afin que la fonctionnalité d'importation des volumes puisse les distinguer.

Si le `azure-netapp-files` ou `gcp-cvs` pilote utilisé, utilisez la commande `tridentctl import volume <backend-name> <volume path> -f /path/pvc.yaml` Pour importer le volume dans Kubernetes qui sera géré par Astra Trident. Cela garantit une référence de volume unique.

À l'exécution de la commande ci-dessus, Astra Trident trouve le volume sur le back-end et lit sa taille. Il ajoute automatiquement (et remplace si nécessaire) la taille du volume du volume du volume persistant configuré. Astra Trident crée ensuite le nouveau volume persistant, et Kubernetes lie la demande de volume persistant au volume persistant.

Lorsqu'un conteneur a été déployé de façon à ce qu'il ait besoin de la demande de volume persistant importée spécifique, il resterait dans un état en attente jusqu'à ce que la paire PVC/PV soit liée via le processus d'importation de volume. Une fois la paire PVC/PV liée, le conteneur doit s'installer, à condition qu'il n'y ait pas d'autres problèmes.

Le déploiement des services OpenShift

Les services de cluster à valeur ajoutée OpenShift offrent des fonctionnalités importantes aux administrateurs de clusters et aux applications hébergées. Le stockage utilisé par ces services peut être provisionné à l'aide des ressources locales. Toutefois, la capacité, la performance, la récupération et la durabilité du service sont souvent limitées. En tirant parti d'une baie de stockage d'entreprise pour fournir la capacité nécessaire à ces services, nous pouvons obtenir un service considérablement amélioré. Cependant, comme pour toutes les applications, OpenShift et les administrateurs de stockage doivent travailler en étroite collaboration afin de déterminer les options les plus adaptées à chacun d'entre eux. La documentation Red Hat doit être largement exploitée pour déterminer les exigences et s'assurer que les besoins en matière de dimensionnement et de performances sont satisfaits.

Service de registre

Le déploiement et la gestion du stockage pour le registre ont été documentés sur "netapp.io" dans le "Blog".

Service de journalisation

Comme les autres services OpenShift, le service de journalisation est déployé avec Ansible, avec les paramètres de configuration fournis par le fichier d'inventaire, également appelé hôtes, fournis avec le manuel de vente. Deux méthodes d'installation sont proposées : le déploiement de la journalisation lors de l'installation initiale d'OpenShift et le déploiement de la journalisation une fois OpenShift installé.

> À partir de la version 3.9 de Red Hat OpenShift, la documentation officielle recommande à NFS d'utiliser le service de journalisation en raison de problèmes de corruption des données. Ceci est basé sur les tests Red Hat de leurs produits. Le serveur NFS d'ONTAP ne présente pas ces problèmes et peut facilement être à nouveau déployé en environnements de journalisation. En fin de compte, le choix du protocole pour le service de journalisation constitue un bon choix. Il suffit de savoir que les deux fonctionneront bien avec les plateformes NetApp. Il n'y a aucune raison d'éviter NFS si c'est votre choix.

Si vous choisissez d'utiliser NFS avec le service de journalisation, vous devez définir la variable Ansible `openshift_enable_unsupported_configurations` à `true` pour éviter que le programme d'installation ne tombe en panne.

Commencez

Le service de journalisation peut, éventuellement, être déployé pour les deux applications ainsi que pour les opérations de base du cluster OpenShift. Si vous choisissez de déployer la journalisation des opérations, en spécifiant la variable `openshift_logging_use_ops` comme `true`, deux instances du service seront créées. Les variables qui contrôlent l'instance de journalisation des opérations contiennent des "OPS", alors que l'instance des applications ne le fait pas.

La configuration des variables Ansible selon la méthode de déploiement est importante afin de s'assurer que le stockage approprié est utilisé par les services sous-jacents. Examinons les options de chacune des méthodes de déploiement.

Les tableaux ci-dessous contiennent uniquement les variables pertinentes pour la configuration du stockage car elles concernent le service de journalisation. Vous trouverez d'autres options dans "Documentation de journalisation Red Hat OpenShift" quels domaines doivent être examinés, configurés et utilisés en fonction de votre déploiement ?

Les variables du tableau ci-dessous entraînent la création d'un volume persistant et de demande de volume persistant pour le service de journalisation à l'aide des informations fournies. Cette méthode est beaucoup moins flexible qu'avec le manuel d'installation des composants après l'installation d'OpenShift. Toutefois, si des volumes sont déjà disponibles, il s'agit d'une option.

Variable	Détails
`openshift_logging_storage_kind`	Réglez sur `nfs` Pour que le programme d'installation crée un volume persistant NFS pour le service de journalisation.
`openshift_logging_storage_host`	Le nom d'hôte ou l'adresse IP de l'hôte NFS. Il doit être défini sur la LIF de données pour votre machine virtuelle.
`openshift_logging_storage_nfs_directory`	Chemin de montage pour l'exportation NFS. Par exemple, si le volume est relié par jonction `/openshift_logging`, vous utiliserez ce chemin pour cette variable.
`openshift_logging_storage_volume_name`	Le nom, par exemple `pv_ose_logs`, De la PV à créer.
`openshift_logging_storage_volume_size`	Taille de l'exportation NFS, par exemple `100Gi`.

Si votre cluster OpenShift est déjà en cours d'exécution et que Trident a donc été déployé et configuré, le programme d'installation peut utiliser le provisionnement dynamique pour créer les volumes. Les variables suivantes doivent être configurées.

Variable	Détails
`openshift_logging_es_pvc_dynamic`	Définis sur true pour l'utilisation de volumes provisionnés dynamiquement.
`openshift_logging_es_pvc_storage_class_name`	Nom de la classe de stockage qui sera utilisée dans le PVC.
`openshift_logging_es_pvc_size`	Taille du volume demandé dans la demande de volume persistant.
`openshift_logging_es_pvc_prefix`	Préfixe pour les ESV utilisés par le service de journalisation.
`openshift_logging_es_ops_pvc_dynamic`	Réglez sur `true` utilisation de volumes provisionnés dynamiquement pour l'instance de journalisation des opérations.
`openshift_logging_es_ops_pvc_storage_class_name`	Nom de la classe de stockage de l'instance de journalisation OPS.
`openshift_logging_es_ops_pvc_size`	Taille de la demande de volume pour l'instance OPS.
`openshift_logging_es_ops_pvc_prefix`	Préfixe pour les ESV de l'instance OPS.

Déploiement de la pile de consignation

Si vous déployez la connexion dans le cadre du processus d'installation initiale d'OpenShift, il vous suffit de suivre le processus de déploiement standard. Ansible configure et déploie les services et les objets OpenShift nécessaires, de sorte que le service soit disponible dès qu'Ansible se termine.

Cependant, si vous déployez après l'installation initiale, vous devez utiliser le PlayBook des composants Ansible. Ce processus peut légèrement évoluer avec différentes versions d'OpenShift, c'est pourquoi nous vous invitons à le lire et à le suivre "Documentation Red Hat OpenShift Container Platform 3.11" pour votre version.

Services de metrics

Le service de metrics fournit à l'administrateur des informations précieuses sur l'état, l'utilisation des ressources et la disponibilité du cluster OpenShift. Il est également nécessaire d'utiliser la fonctionnalité de montée en charge automatique des pods. De nombreuses entreprises utilisent les données issues du service de metrics pour leurs applications de refacturation et/ou de show-back.

Comme pour le service de journalisation, OpenShift dans son ensemble, Ansible est utilisé pour déployer le service de metrics. De même, comme le service de journalisation, le service de metrics peut être déployé lors d'une configuration initiale du cluster ou après son fonctionnement à l'aide de la méthode d'installation du composant. Les tableaux suivants contiennent les variables importantes lors de la configuration du stockage persistant pour le service de metrics.

> Les tableaux ci-dessous contiennent uniquement les variables pertinentes pour la configuration du stockage car elles concernent le service de metrics. De nombreuses autres options sont disponibles dans la documentation qui doit être examinée, configurée et utilisée en fonction de votre déploiement.

Variable	Détails
`openshift_metrics_storage_kind`	Réglez sur `nfs` Pour que le programme d'installation crée un volume persistant NFS pour le service de journalisation.
`openshift_metrics_storage_host`	Le nom d'hôte ou l'adresse IP de l'hôte NFS. Il doit être défini sur la LIF de données pour votre SVM.
`openshift_metrics_storage_nfs_directory`	Chemin de montage pour l'exportation NFS. Par exemple, si le volume est relié par jonction `/openshift_metrics`, vous utiliserez ce chemin pour cette variable.
`openshift_metrics_storage_volume_name`	Le nom, par exemple `pv_ose_metrics`, De la PV à créer.
`openshift_metrics_storage_volume_size`	Taille de l'exportation NFS, par exemple `100Gi`.

Si votre cluster OpenShift est déjà en cours d'exécution et que Trident a donc été déployé et configuré, le programme d'installation peut utiliser le provisionnement dynamique pour créer les volumes. Les variables suivantes doivent être configurées.

Variable	Détails
`openshift_metrics_cassandra_pvc_prefix`	Préfixe à utiliser pour les ESV de metrics.

Variable	Détails
`openshift_metrics_cassandra_pvc_size`	Taille des volumes à demander.
`openshift_metrics_cassandra_storage_type`	Le type de stockage à utiliser pour les metrics, doit être défini sur dynamique pour qu'Ansible crée des demandes de volume persistant avec la classe de stockage appropriée.
`openshift_metrics_cassanda_pvc_storage_class_name`	Nom de la classe de stockage à utiliser.

Déployez le service de metrics

Déployez le service à l'aide des variables Ansible appropriées définies dans votre fichier hôtes/d'inventaire. Si vous déployez au moment de l'installation d'OpenShift, le volume persistant est créé et utilisé automatiquement. Si vous déployez l'utilisation des playbooks, après l'installation d'OpenShift, Ansible crée toutes les demandes de volume persistant nécessaires et, après que Astra Trident a provisionné le stockage pour eux, déployez le service.

Les variables ci-dessus et le processus de déploiement peuvent changer avec chaque version d'OpenShift. Vérifiez et suivez "Guide de déploiement OpenShift de Red Hat" pour votre version afin qu'elle soit configurée pour votre environnement.

Protection des données

Découvrez les options de protection des données et de restauration fournies par les plateformes de stockage NetApp. Astra Trident peut provisionner des volumes qui peuvent bénéficier de certaines de ces fonctionnalités. Vous devez disposer d'une stratégie de protection et de restauration des données pour chaque application ayant des exigences de persistance.

Sauvegardez le `etcd` données de cluster

Astra Trident stocke ses métadonnées dans le cluster Kubernetes `etcd` base de données. Sauvegarder régulièrement le `etcd` Les données en cluster sont importantes pour la restauration de clusters Kubernetes en cas d'incident.

Étapes

1. Le `etcdctl snapshot save` vous permet de créer un snapshot instantané de l' `etcd` cluster :

```
sudo docker run --rm -v /backup:/backup \
  --network host \
  -v /etc/kubernetes/pki/etcd:/etc/kubernetes/pki/etcd \
  --env ETCDCTL_API=3 \
  registry.k8s.io/etcd-amd64:3.2.18 \
  etcdctl --endpoints=https://127.0.0.1:2379 \
  --cacert=/etc/kubernetes/pki/etcd/ca.crt \
  --cert=/etc/kubernetes/pki/etcd/healthcheck-client.crt \
  --key=/etc/kubernetes/pki/etcd/healthcheck-client.key \
  snapshot save /backup/etcd-snapshot.db
```

Cette commande crée un snapshot ETCD en faisant tourner un conteneur ETCD et le sauvegarde dans le `/backup` répertoire.

2. En cas d'incident, vous pouvez activer un cluster Kubernetes à l'aide des snapshots ETCD. Utilisez le `etcdctl snapshot restore` commande permettant de restaurer un snapshot spécifique pris sur le `/var/lib/etcd` dossier. Après la restauration, vérifiez si `/var/lib/etcd` le dossier a été rempli avec le `member` dossier. Voici un exemple de `etcdctl snapshot restore` commande :

```
etcdctl snapshot restore '/backup/etcd-snapshot-latest.db' ; mv
/default.etcd/member/ /var/lib/etcd/
```

3. Avant d'initialiser le cluster Kubernetes, copiez tous les certificats nécessaires.
4. Créez le cluster avec le `--ignore-preflight-errors=DirAvailable—var-lib-etcd` drapeau.
5. Une fois le cluster lancé, assurez-vous que les modules du système kube ont démarré.
6. Utilisez le `kubectl get crd` Commande pour vérifier si les ressources personnalisées créées par Trident sont présentes et récupérer des objets Trident afin de s'assurer que toutes les données sont disponibles.

Récupérer les données à l'aide des snapshots ONTAP

Les snapshots jouent un rôle important en proposant des options de restauration ponctuelles pour les données d'application. Toutefois, les snapshots ne sont pas des sauvegardes en elles-mêmes, elles ne protègent pas contre les défaillances du système de stockage ou autres catastrophes. Cependant, ils constituent un moyen pratique, rapide et simple de restaurer des données dans la plupart des scénarios. Découvrez comment utiliser la technologie Snapshot de ONTAP pour sauvegarder les volumes et les restaurer.

- Si la politique de snapshot n'a pas été définie dans le backend, elle utilise par défaut le `none` politique. Résultat : ONTAP ne prend pas de snapshots automatiques. Toutefois, l'administrateur du stockage peut effectuer manuellement des snapshots ou modifier la règle Snapshot via l'interface de gestion ONTAP. Cela n'affecte pas le fonctionnement de Trident.
- Le répertoire de snapshot est masqué par défaut. Cela facilite une compatibilité maximale des volumes provisionnés à l'aide de `ontap-nas` et `ontap-nas-economy` pilotes. Activez le `.snapshot` répertoire lors de l'utilisation du `ontap-nas` et `ontap-nas-economy` pilotes permettant aux applications de récupérer directement les données à partir des snapshots.
- Restaurer un volume à un état enregistré dans un instantané précédent à l'aide du `volume snapshot`

`restore` Commande CLI ONTAP. Lorsque vous restaurez une copie snapshot, l'opération de restauration écrase la configuration de volume existante. Toute modification apportée aux données du volume après la création de la copie Snapshot est perdue.

```
cluster1::*> volume snapshot restore -vserver vs0 -volume vol3 -snapshot
vol3_snap_archive
```

Réplication des données à l'aide de ONTAP

La réplication des données peut jouer un rôle important dans la protection contre les pertes de données dues à une défaillance de la baie de stockage.

Pour en savoir plus sur les technologies de réplication ONTAP, consultez le "Documentation ONTAP".

Réplication des machines virtuelles de stockage (SVM) SnapMirror

Vous pouvez utiliser "SnapMirror" pour répliquer un SVM complet, qui inclut ses paramètres de configuration et ses volumes. En cas d'incident, vous pouvez activer la SVM de destination SnapMirror pour démarrer le service des données. Vous pouvez revenir au primaire lorsque les systèmes sont restaurés.

Astra Trident ne peut pas configurer lui-même les relations de réplication. L'administrateur du stockage peut donc utiliser la fonctionnalité de réplication du SVM SnapMirror d'ONTAP pour répliquer automatiquement les volumes vers une destination de reprise après incident.

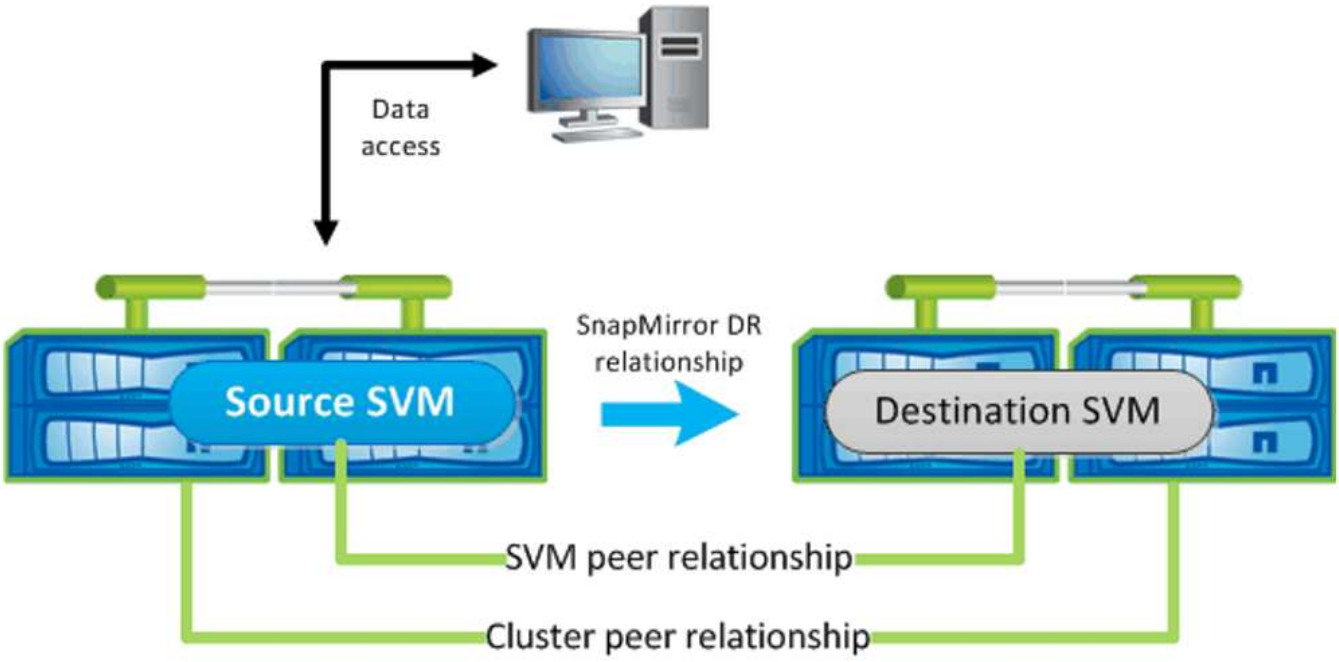
Envisagez la commande suivante si vous prévoyez d'utiliser la fonctionnalité de réplication SVM SnapMirror ou si vous utilisez actuellement la fonctionnalité :

- Vous devez créer un système back-end distinct pour chaque SVM, sur lequel SVM-DR est activé.
- Vous devez configurer les classes de stockage de manière à ne pas sélectionner les systèmes back-end répliqués, sauf lorsque cela est nécessaire. Cela est important pour éviter la mise en service des volumes qui ne nécessitent pas de protection de la relation de réplication sur les back-end compatibles avec SVM-DR.
- Les administrateurs d'applications doivent comprendre les coûts et la complexité supplémentaires liés à la réplication des données, et un plan de restauration doit être déterminé avant d'exploiter la réplication des données.
- Avant d'activer la SVM de destination SnapMirror, arrêter tous les transferts SnapMirror planifiés, abandonner tous les transferts SnapMirror en cours, interrompre la relation de réplication, arrêter la SVM source puis démarrer la SVM de destination SnapMirror.
- Astra Trident ne détecte pas automatiquement les défaillances du SVM. Par conséquent, en cas d'échec, l'administrateur doit exécuter le `tridentctl backend update` Commande permettant de déclencher le basculement de Trident vers le nouveau back-end.

Voici une présentation des étapes de configuration des SVM :

- Configurer le peering entre le cluster source et destination et SVM
- Créer le SVM de destination à l'aide de l' `-subtype dp-destination` option.
- Créez une planification de tâches de réplication afin de vous assurer que la réplication se déroule aux intervalles requis.

- Créer une réplication SnapMirror depuis le SVM de destination vers le SVM source à l'aide de `-identity -preserve true` Option pour s'assurer que les configurations du SVM source et les interfaces du SVM source sont copiées vers la destination. Depuis le SVM de destination, initialiser la relation de réplication SVM SnapMirror



Workflow de reprise d'activité pour Trident

Astra Trident 19.07 et les versions ultérieures utilisent des CRD Kubernetes pour stocker et gérer son propre état. Elle utilise celle du cluster Kubernetes `etcd` pour stocker ses métadonnées. On suppose ici que Kubernetes `etcd` Les fichiers de données et les certificats sont stockés sur NetApp FlexVolume. Ce FlexVolume réside dans un SVM, qui dispose d'une relation SVM-DR SnapMirror avec un SVM de destination sur le site secondaire.

La procédure suivante décrit comment restaurer un cluster Kubernetes maître avec Astra Trident en cas d'incident :

1. En cas de défaillance du SVM source, activer le SVM de destination SnapMirror Pour cela, il faut arrêter des transferts SnapMirror planifiés, abandonner les transferts SnapMirror en cours, interrompre la relation de réplication, arrêter la SVM source et démarrer la SVM de destination.
2. Depuis le SVM de destination, montez le volume qui contient l'environnement Kubernetes `etcd` fichiers de données et certificats sur l'hôte qui seront configurés en tant que nœud maître.
3. Copiez tous les certificats requis se rapportant au cluster Kubernetes sous `/etc/kubernetes/pki` et le etcd `member` fichiers sous `/var/lib/etcd`.
4. Créez un cluster Kubernetes en utilisant le `kubeadm init` commande avec `--ignore-preflight -errors=DirAvailable—var-lib-etcd` drapeau. Les noms d'hôte utilisés pour les nœuds Kubernetes doivent être identiques au cluster Kubernetes source.
5. Exécutez le `kubectl get crd` Commande pour vérifier si toutes les ressources personnalisées Trident ont été extraites et récupérer les objets Trident pour vérifier que toutes les données sont disponibles.
6. Mise à jour de tous les systèmes back-end requis pour refléter le nouveau nom de SVM de destination en exécutant la `./tridentctl update backend <backend-name> -f <backend-json-file> -n`

`<namespace>` commande.

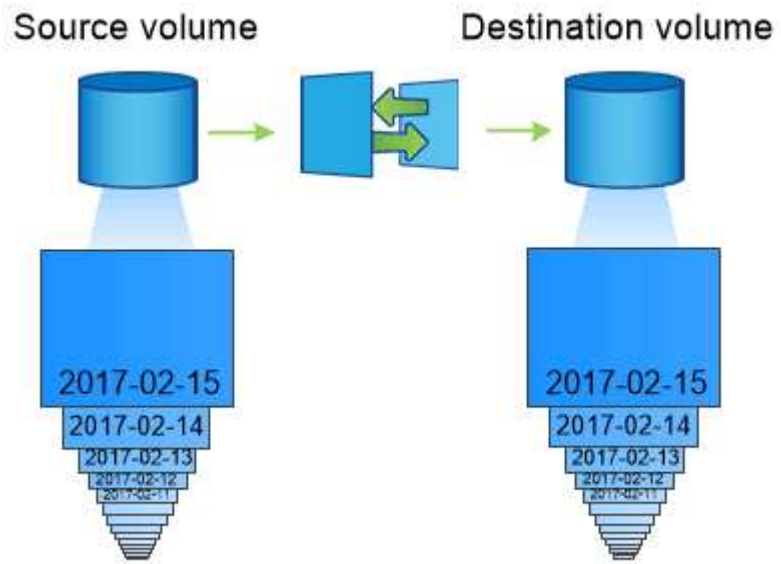
> Lorsque le SVM de destination est activé pour les volumes persistants des applications, tous les volumes provisionnés par Trident commencent à transmettre les données. Une fois le cluster Kubernetes configuré sur le système de destination conformément aux étapes décrites ci-dessus, tous les déploiements et les pods sont démarrés et les applications conteneurisées doivent s'exécuter sans aucun problème.

Réplication de volume SnapMirror

La réplication de volume ONTAP SnapMirror est une fonctionnalité de reprise d'activité qui permet le basculement vers le stockage de destination à partir d'un stockage primaire au niveau des volumes. SnapMirror crée une réplique de volume ou un miroir du stockage primaire sur le stockage secondaire en synchronisant les snapshots.

Voici une synthèse des étapes de configuration de la réplication de volume ONTAP SnapMirror :

- Configurez le peering entre les clusters dans lesquels les volumes résident et les SVM qui fournissent les données des volumes.
- Créer une règle SnapMirror, qui contrôle le comportement de la relation et spécifie les attributs de configuration pour cette relation.
- Créez une relation SnapMirror entre le volume de destination et le volume source à l'aide de la[`snapmirror create` commande^] et attribuez la règle SnapMirror appropriée.
- Une fois la relation SnapMirror créée, initialisez la relation pour qu'un transfert de base du volume source vers le volume de destination soit terminé.



Workflow de reprise d'activité de volumes SnapMirror pour Trident

La procédure suivante décrit comment restaurer un cluster Kubernetes maître avec Astra Trident.

1. En cas d'incident, arrêter tous les transferts SnapMirror programmés et abandonner tous les transferts SnapMirror en cours. Rompez la relation de réplication entre les volumes de destination et source de sorte que le volume de destination soit lu/écrit.

2. Depuis le SVM de destination, montez le volume qui contient l'environnement Kubernetes `etcd` fichiers de données et certificats sur l'hôte, qui sera configuré en tant que nœud maître.
3. Copiez tous les certificats requis se rapportant au cluster Kubernetes sous `/etc/kubernetes/pki` et le etcd `member` fichiers sous `/var/lib/etcd`.
4. Créez un cluster Kubernetes en exécutant le `kubeadm init` commande avec `--ignore-preflight -errors=DirAvailable—var-lib-etcd` drapeau. Les noms d'hôte doivent être identiques au cluster Kubernetes source.
5. Exécutez le `kubectl get crd` Commande pour vérifier si toutes les ressources personnalisées Trident ont été extraites et récupérer des objets Trident pour s'assurer que toutes les données sont disponibles.
6. Nettoyez les systèmes back-end précédents et créez de nouveaux systèmes back-end sur Trident. Préciser la nouvelle LIF de management, le nouveau nom du SVM et le mot de passe du SVM de destination.

Workflow de reprise d'activité pour les volumes persistants des applications

Les étapes suivantes décrivent comment mettre à disposition les volumes de destination SnapMirror pour les workloads conteneurisés en cas d'incident :

1. Arrêt de tous les transferts SnapMirror programmés et abandon de tous les transferts SnapMirror en cours. Rompez la relation de réplication entre le volume de destination et le volume source pour que le volume de destination devienne read/write. Nettoyer les déploiements qui consomtaient du volume persistant lié aux volumes sur la SVM source.
2. Une fois le cluster Kubernetes configuré sur le côté destination, suivez les étapes décrites ci-dessus pour nettoyer les déploiements, les demandes de volume persistant et le volume persistant à partir du cluster Kubernetes.
3. Créer de nouveaux systèmes back-end sur Trident en spécifiant la nouvelle LIF de gestion et de données, un nouveau nom de SVM et un nouveau mot de passe du SVM de destination.
4. Importez les volumes requis en tant que volume persistant lié à une nouvelle demande de volume persistant à l'aide de la fonctionnalité d'importation Trident.
5. Redéployez les déploiements d'applications avec les demandes de volume nouvellement créées.

Restaurez les données à l'aide des snapshots Element

Sauvegardez les données sur un volume Element en définissant une planification Snapshot pour le volume. Vous pouvez ainsi vérifier que les snapshots sont effectués à intervalles réguliers. Vous devez définir la planification des snapshots à l'aide de l'interface utilisateur ou des API d'Element. Actuellement, il n'est pas possible de définir un planning de snapshots sur un volume via la `solidfire-san` conducteur.

En cas de corruption des données, vous pouvez choisir un snapshot en particulier et restaurer manuellement le volume vers le Snapshot à l'aide de l'interface utilisateur ou des API Element. Cette opération rétablit les modifications apportées au volume depuis la création du snapshot.

Sécurité

Sécurité

Assurez-vous que l'installation d'Astra Trident est sécurisée à l'aide des recommandations indiquées ici.

Exécutez Astra Trident dans son propre espace de noms

Il est important d'empêcher les applications, les administrateurs d'applications, les utilisateurs et les applications de gestion d'accéder aux définitions d'objets Astra Trident ou aux pods pour assurer un stockage fiable et bloquer tout risque d'activité malveillante.

Pour séparer les autres applications et utilisateurs d'Astra Trident, installez toujours Astra Trident dans son propre espace de noms Kubernetes (`trident`). L'utilisation d'Astra Trident dans son propre espace de noms garantit que seul le personnel d'administration Kubernetes a accès au pod Trident Astra et aux artéfacts (tels que les secrets d'arrière-plan et CHAP le cas échéant) stockés dans les objets CRD devant être namestes. Vous devez vous assurer que seuls les administrateurs ont accès à l'espace de noms Astra Trident et y ont donc accès `tridentctl` client supplémentaire.

Utilisez l'authentification CHAP avec les systèmes back-end ONTAP SAN

Astra Trident prend en charge l'authentification CHAP pour les workloads SAN de ONTAP (à l'aide du `ontap-san` et `ontap-san-economy` pilotes). NetApp recommande d'utiliser le protocole CHAP bidirectionnel avec Astra Trident pour l'authentification entre l'hôte et le système back-end de stockage.

Pour les systèmes ONTAP back-end qui utilisent les pilotes de stockage SAN, Astra Trident peut configurer le protocole CHAP bidirectionnel et gérer les noms d'utilisateur et les secrets CHAP via `tridentctl`. Voir "ici" Pour comprendre comment Astra Trident configure le protocole CHAP sur les systèmes back-end ONTAP.

> La prise en charge CHAP pour les systèmes back-end ONTAP est disponible avec Trident 20.04 et versions ultérieures.

Utilisez l'authentification CHAP avec les systèmes back-end NetApp HCI et SolidFire

NetApp recommande de déployer le protocole CHAP bidirectionnel pour garantir l'authentification entre l'hôte et les systèmes back-end NetApp HCI et SolidFire. Astra Trident utilise un objet secret qui inclut deux mots de passe CHAP par locataire. Lorsque Trident est installé en tant que fournisseur CSI, il gère les secrets CHAP et les stocke dans un `tridentvolume` Objet CR pour la PV correspondante. Lorsque vous créez un volume persistant, CSI Trident utilise les secrets CHAP pour initier une session iSCSI et communiquer avec le système NetApp HCI et SolidFire via CHAP.

> Les volumes créés par CSI Trident ne sont associés à aucun groupe d'accès de volume.

Sur le système front-end non CSI, la connexion de volumes en tant que périphériques sur les nœuds workers est gérée par Kubernetes. Après la création de volumes, Astra Trident effectue un appel d'API vers le système NetApp HCI/SolidFire pour récupérer les secrets de ce locataire n'existe pas encore. Astra Trident transmet ensuite les secrets de Kubernetes. Le kubelet situé sur chaque nœud accède aux secrets de l'API Kubernetes et les utilise pour exécuter/activer CHAP entre chaque nœud accédant au volume et le système NetApp HCI/SolidFire où se trouvent les volumes.

Utilisez Astra Trident avec NVE et NAE

NetApp ONTAP assure le chiffrement des données au repos pour protéger les données sensibles en cas de vol, de retour ou de reconversion d'un disque. Pour plus de détails, reportez-vous à "Configurer la présentation de NetApp Volume Encryption".

- Si NAE est activé sur le back-end, tous les volumes provisionnés dans Astra Trident seront NAE.
- Si NAE n'est pas activé sur le back-end, les volumes provisionnés dans Astra Trident seront compatibles avec NVE, à moins que vous n'ayez défini le indicateur de chiffrement NVE sur `false` en configuration

back-end.

> Les volumes créés dans Astra Trident sur un système back-end compatible NAE doivent être chiffrés NVE ou NAE.
>
> - Vous pouvez définir l'indicateur de chiffrement NVE sur `true` Dans la configuration back-end Trident pour remplacer le chiffrement NAE et utiliser une clé de chiffrement spécifique sur la base du volume.
> - Définition de l'indicateur de chiffrement NVE sur `false` Sur un système back-end NAE, un volume basé sur NAE est créé. Vous ne pouvez pas désactiver le chiffrement NAE en définissant l'indicateur de chiffrement NVE sur `false`.

- Vous pouvez créer manuellement un volume NVE dans Astra Trident en définissant explicitement l'indicateur de chiffrement NVE sur `true`.

Pour plus d'informations sur les options de configuration du back-end, reportez-vous à :

- "Options de configuration du SAN ONTAP"
- "Options de configuration du stockage NAS ONTAP"

Configuration de clé unifiée Linux (LUKS)

Vous pouvez activer l'utilitaire Linux Unified Key Setup (LUKS) pour chiffrer les volumes SAN ONTAP et SAN ONTAP ÉCONOMIQUES sur Astra Trident. Astra Trident prend en charge la rotation de phrase secrète et l'extension de volume pour les volumes chiffrés LUKS.

Dans Astra Trident, les volumes chiffrés LUKS utilisent le sypher et le mode aes-xts-m64, comme recommandé par "NIST".

Avant de commencer

- Les nœuds worker doivent avoir cryptsetup 2.1 ou supérieur (mais inférieur à 3.0) installé. Pour plus d'informations, rendez-vous sur "Gitlab : cryptsetup".
- Pour des raisons de performances, nous recommandons aux nœuds workers de prendre en charge les nouvelles instructions AES-ni (Advanced Encryption Standard New instructions). Pour vérifier la prise en charge AES-ni, exécutez la commande suivante :

```
grep "aes" /proc/cpuinfo
```

Si rien n'est renvoyé, votre processeur ne prend pas en charge AES-ni. Pour plus d'informations sur AES-ni, visitez le site : "Intel : instructions AES-ni (Advanced Encryption Standard instructions)".

Activez le cryptage LUKS

Vous pouvez activer le chiffrement côté hôte par volume en utilisant Linux Unified Key Setup (LUKS) pour SAN ONTAP et les volumes ÉCONOMIQUES SAN ONTAP.

Étapes

1. Définissez les attributs de cryptage LUKS dans la configuration back-end. Pour plus d'informations sur les options de configuration des back-end pour SAN ONTAP, reportez-vous à "Options de configuration du SAN ONTAP".

```
"storage": [
    {
        "labels":{"luks": "true"},
        "zone":"us_east_1a",
        "defaults": {
            "luksEncryption": "true"
        }
    },
    {
        "labels":{"luks": "false"},
        "zone":"us_east_1a",
        "defaults": {
            "luksEncryption": "false"
        }
    },
]
```

2. Utiliser `parameters.selector` Pour définir les pools de stockage à l'aide du cryptage LUKS. Par exemple :

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: luks
provisioner: netapp.io/trident
parameters:
  selector: "luks=true"
  csi.storage.k8s.io/node-stage-secret-name: luks-${pvc.name}
  csi.storage.k8s.io/node-stage-secret-namespace: ${pvc.namespace}
```

3. Créez un secret qui contient la phrase de passe LUKS. Par exemple :

```
kubectl -n trident create -f luks-pvc1.yaml
apiVersion: v1
kind: Secret
metadata:
  name: luks-pvc1
stringData:
  luks-passphrase-name: A
  luks-passphrase: secretA
```

Limites

Les volumes LUKS-chiffrés ne peuvent pas tirer parti de la déduplication et de la compression ONTAP.

Faites pivoter une phrase de passe LUKS

Vous pouvez faire pivoter la phrase de passe LUKS et confirmer la rotation.

> N'oubliez pas une phrase de passe tant que vous n'avez pas vérifié qu'elle n'est plus référencée par un volume, un snapshot ou un secret. En cas de perte d'une phrase secrète référencée, vous risquez de ne pas pouvoir monter le volume et les données resteront cryptées et inaccessibles.

Description de la tâche

La rotation de la phrase de passe LUKS se produit lorsqu'un pod qui monte le volume est créé après la spécification d'une nouvelle phrase de passe LUKS. Lors de la création d'un pod, Astra Trident compare la phrase de passe LUKS sur le volume à la phrase de passe active dans le secret.

- Si la phrase de passe du volume ne correspond pas à la phrase de passe active dans le secret, la rotation se produit.
- Si la phrase de passe du volume correspond à la phrase de passe active dans le secret, le `previous-luks-passphrase` paramètre ignoré.

Étapes

1. Ajoutez le `node-publish-secret-name` et `node-publish-secret-namespace` Paramètres de classe de stockage. Par exemple :

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: csi-san
provisioner: csi.trident.netapp.io
parameters:
  trident.netapp.io/backendType: "ontap-san"
  csi.storage.k8s.io/node-stage-secret-name: luks
  csi.storage.k8s.io/node-stage-secret-namespace: ${pvc.namespace}
  csi.storage.k8s.io/node-publish-secret-name: luks
  csi.storage.k8s.io/node-publish-secret-namespace: ${pvc.namespace}
```

2. Identifier les phrases de passe existantes sur le volume ou l'instantané.

Volumétrie

```
tridentctl -d get volume luks-pvc1
GET http://127.0.0.1:8000/trident/v1/volume/<volumeID>

...luksPassphraseNames:["A"]
```

Snapshot

```
tridentctl -d get snapshot luks-pvc1
GET http://127.0.0.1:8000/trident/v1/volume/<volumeID>/<snapshotID>

...luksPassphraseNames:["A"]
```

3. Mettez à jour le secret LUKS pour le volume afin de spécifier les phrases de passe nouvelles et précédentes. Bien sûr `previous-luke-passphrase-name` et `previous-luks-passphrase` faites correspondre la phrase de passe précédente.

```
apiVersion: v1
kind: Secret
metadata:
  name: luks-pvc1
stringData:
  luks-passphrase-name: B
  luks-passphrase: secretB
  previous-luks-passphrase-name: A
  previous-luks-passphrase: secretA
```

4. Créez un nouveau pod qui monte le volume. Ceci est nécessaire pour lancer la rotation.
5. Vérifiez que la phrase de passe a été pivotée.

Volumétrie

```
tridentctl -d get volume luks-pvc1
GET http://127.0.0.1:8000/trident/v1/volume/<volumeID>

...luksPassphraseNames:["B"]
```

Snapshot

```
tridentctl -d get snapshot luks-pvc1
GET http://127.0.0.1:8000/trident/v1/volume/<volumeID>/<snapshotID>

...luksPassphraseNames:["B"]
```

Résultats

La phrase de passe a été pivotée lorsque seule la nouvelle phrase de passe est renvoyée sur le volume et le snapshot.

Si deux phrases de passe sont renvoyées, par exemple `luksPassphraseNames: ["B", "A"]`, la rotation est incomplète. Vous pouvez déclencher un nouveau pod pour tenter de terminer la rotation.

Activer l'extension de volume

Vous pouvez activer l'extension de volume sur un volume chiffré LUKS.

Étapes

1. Activez le `CSINodeExpandSecret` feature gate (bêta 1.25+). Reportez-vous à la section "Kubernetes 1.25 : utilisez les secrets de l'extension des volumes CSI basée sur des nœuds" pour plus d'informations.
2. Ajoutez le `node-expand-secret-name` et `node-expand-secret-namespace` Paramètres de classe de stockage. Par exemple :

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: luks
provisioner: netapp.io/trident
parameters:
  selector: "luks=true"
  csi.storage.k8s.io/node-stage-secret-name: luks-${pvc.name}
  csi.storage.k8s.io/node-stage-secret-namespace: ${pvc.namespace}
  csi.storage.k8s.io/node-expand-secret-name: luks-${pvc.name}
  csi.storage.k8s.io/node-expand-secret-namespace: ${pvc.namespace}
allowVolumeExpansion: true
```

Résultats

Lorsque vous initiez l'extension du stockage en ligne, le kubelet transmet les identifiants appropriés au pilote.

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	ontap-nas, ontap-san, solidfire-san, azure-netapp-files, gcp-cvs ontap-san-économie
trident.netapp.io/splitOnClone	SplitOnClone	ontap-nas, ontap-san
trident.netapp.io/protocol	protocole	toutes
trident.netapp.io/exportPolicy	ExportPolicy	ontap-nas, économie ontap-nas, ontap-nas-flexgroup
trident.netapp.io/snapshotPolicy	Politique de snapshots	ontap-nas, ontap-nas-économie, ontap-nas-flexgroup, ontap-san
trident.netapp.io/snapshotReserve	Réserve de snapshots	ontap-nas, ontap-nas-flexgroup, ontap-san, gcp-cvs
trident.netapp.io/snapshotDirectory	Répertoire de snapshots	ontap-nas, économie ontap-nas, ontap-nas-flexgroup
trident.netapp.io/unixPermissions	Autorisations unix	ontap-nas, économie ontap-nas, ontap-nas-flexgroup
trident.netapp.io/blockSize	Taille de bloc	solidfire-san

Si le volume persistant créé est 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 version antérieure) 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	Mappage des noms backend avec les listes de pools de stockage dans
Des médutiquesde stockage	Mapper[string]StringList	non	Mappage des noms backend avec les listes de pools de stockage dans
Exclus du stockagePools	Mapper[string]StringList	non	Mappage des noms backend avec les listes de pools de stockage dans

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

Attributs de sélection du pool de stockage

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

Attribut	Type	Valeurs	Offre	Demande	Pris en charge par
support[1]	chaîne	hdd, hybride, ssd	Le pool contient des supports de ce type ; hybride signifie les deux	Type de support spécifié	ontap-nas, ontap-nas-économie, ontap-nas-flexgroup, ontap-san, solidfire-san

Attribut	Type	Valeurs	Offre	Demande	Pris en charge par
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	Version Kubernetes
Fstype	chaîne	ext4, ext3, xfs, etc	Type de système de fichiers pour les volumes en mode bloc	solidfire-san, ontap-nas, ontap-nas-économie, ontap-nas-flexgroup, ontap-san, ontap-san-économie	Tout
Volumeallowexpansion	booléen	vrai, faux	Activez ou désactivez la prise en charge pour augmenter la taille de la demande de volume persistant	ontap-nas, économie ontap-nas, ontap-nas-flexgroup, ontap-san, ontap-san-économie, solidfire-san, gcp-cvs, azure-netapp-files	1.11+

Attribut	Type	Valeurs	Description	Facteurs pertinents	Version Kubernetes
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
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

Attribut	Type	Obligatoire	Description
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 automatiquement cet objet volume dans le cadre du provisionnement.

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 »)

Attribut	Type	Obligatoire	Description
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 renseignées en obtenant les détails du volume de sauvegarde.

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` drapeaux: * `-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]
```

Drapeaux: * `-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]
```

Drapeaux: * `-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]
```

Drapeaux: * `--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 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`

Seccomp	Le profil Seccomp est toujours « confiné » dans des conteneurs privilégiés. Par conséquent, il ne peut pas être activé sur Astra Trident.

Autorisations	Description
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`

Versions antérieures de la documentation

Si vous n’exécutez pas Astra Trident 23.01, la documentation relative aux versions précédentes est disponible.

- "Astra Trident 22.10"
- "Astra Trident 22.07"
- "Astra Trident 22.04"
- "Astra Trident 22.01"

Mentions légales

Les mentions légales donnent accès aux déclarations de copyright, aux marques, aux brevets, etc.

Droits d’auteur

"https://www.netapp.com/company/legal/copyright/"

Marques déposées

NetApp, le logo NETAPP et les marques mentionnées sur la page des marques commerciales NetApp sont des marques commerciales de NetApp, Inc. Les autres noms de sociétés et de produits peuvent être des marques commerciales de leurs propriétaires respectifs.

"https://www.netapp.com/company/legal/trademarks/"

Brevets

Vous trouverez une liste actuelle des brevets appartenant à NetApp à l’adresse suivante :

https://www.netapp.com/pdf.html?item=/media/11887-patentspage.pdf

Politique de confidentialité

"https://www.netapp.com/company/legal/privacy-policy/"

Source ouverte

Vous pouvez consulter les informations de copyright et licences tierces utilisées dans le logiciel NetApp pour Astra Trident dans le fichier d’avis correspondant à chaque version, à l’adresse https://github.com/NetApp/trident/.

Informations sur le copyright

Copyright © 2025 NetApp, Inc. Tous droits réservés. Imprimé aux États-Unis. Aucune partie de ce document protégé par copyright ne peut être reproduite sous quelque forme que ce soit ou selon quelque méthode que ce soit (graphique, électronique ou mécanique, notamment par photocopie, enregistrement ou stockage dans un système de récupération électronique) sans l'autorisation écrite préalable du détenteur du droit de copyright.

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

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

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

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

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

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

Informations sur les marques commerciales

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