Documentation Trident 24.10

Trident

NetApp
November 22, 2024

Sommaire

Documentation Trident 24.10

Notes de mise à jour

Quoi de neuf

Les notes de version fournissent des informations sur les nouvelles fonctionnalités, les améliorations et les correctifs de la dernière version de 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 24.10

Améliorations

- Le pilote Google Cloud NetApp volumes est désormais disponible pour les volumes NFS et prend en charge le provisionnement avec détection de zone.
- L'identité de workload GCP sera utilisée comme identité cloud pour les volumes NetApp de Google Cloud avec GKE.
- Ajout d'un `formatOptions` paramètre de configuration aux pilotes ONTAP-SAN et ONTAP-SAN-Economy pour permettre aux utilisateurs de spécifier les options de format de LUN.
- Taille minimale du volume Azure NetApp Files réduite à 50 Gio. La nouvelle taille minimale d'Azure devrait être globalement disponible en novembre.
- Ajout d'un `denyNewVolumePools` paramètre de configuration pour limiter les pilotes ONTAP-NAS-Economy et ONTAP-SAN-Economy aux pools FlexVol préexistants.
- Détection supplémentaire pour l'ajout, la suppression ou le renommage d'agrégats du SVM sur tous les pilotes ONTAP
- Ajout de la surcharge de 18 Mio aux LUN LUKS pour garantir que la taille de PVC signalée est utilisable.
- Amélioration de la gestion des étapes et des déconnexions des nœuds ONTAP-SAN et ONTAP-SAN-Economy pour permettre le retrait des périphériques après une phase d'échec.
- Ajout d'un générateur de rôles personnalisé permettant aux clients de créer un rôle minimaliste pour Trident dans ONTAP.
- Ajout d'une journalisation supplémentaire pour le dépannage `lsscsi` ("Question no 792").

Kubernetes

- Ajout de nouvelles fonctionnalités Trident pour les workflows natifs Kubernetes :
  - Protection des données
  - Migration des données
  - Reprise après incident
  - Mobilité des applications

  "En savoir plus sur Trident Protect".
- Ajout d'un nouvel indicateur `--k8s_api_qps` aux installateurs pour définir la valeur QPS utilisée par

Trident pour communiquer avec le serveur API Kubernetes.

- Indicateur ajouté `--node-prep` aux programmes d'installation pour la gestion automatique des dépendances des protocoles de stockage sur les nœuds de cluster Kubernetes. Compatibilité testée et vérifiée avec le protocole de stockage iSCSI Amazon Linux 2023
- Ajout de la prise en charge du détachement forcé pour les volumes ONTAP-NAS-Economy dans les scénarios d'arrêt de nœud non normal.
- Les nouveaux volumes NFS ONTAP-NAS-Economy utiliseront des règles d'export par qtree lors de l'utilisation de `autoExportPolicy` l'option backend. Les qtrees ne sont mappés sur des règles d'exportation restrictives du nœud qu'au moment de la publication, afin d'améliorer le contrôle d'accès et la sécurité. Lorsque Trident supprime le volume de tous les nœuds, les qtrees existants sont basculés vers le nouveau modèle de règles d'export pour le faire, sans impact sur les workloads actifs.
- Prise en charge de Kubernetes 1.31.

Améliorations expérimentales

- Ajout d'un aperçu technique de la prise en charge de Fibre Channel sur le pilote ONTAP-SAN. Reportez-vous à la "Prise en charge de Fibre Channel".

Correctifs

- **Kubernetes** :
  - Crochet d'admission fixe de Rancher empêchant les installations de Trident Helm ("Question no 839").
  - Clé d'affinité fixe dans les valeurs du graphique Helm ("Question no 898").
  - Fixed tridentControllerPluginNodeSelector/tridentNodePluginNodeSelector ne fonctionnera pas avec la valeur "true" ("Question no 899").
  - Les snapshots éphémères créés lors du clonage () ont été supprimés"Question no 901".
- Ajout de la prise en charge de Windows Server 2019.
- Fixe `Go mod Tidy` dans Trident repo ("Question no 767").

Dérations

- **Kubernetes:**
  - Mise à jour de la version 1.25 de Kubernetes minimale prise en charge.
  - Suppression de la prise en charge de la stratégie de sécurité POD.

Changement de marque du produit

À partir de la version 24.10, Astra Trident a été renommée Trident (NetApp Trident). Ce changement de marque n'affecte en rien les fonctionnalités, les plateformes prises en charge ou l'interopérabilité pour Trident.

Changements en 24.06

Améliorations

- **IMPORTANT** : le `limitVolumeSize` paramètre limite désormais la taille qtree/LUN dans les pilotes économiques ONTAP. Utilisez le nouveau `limitVolumePoolSize` paramètre pour contrôler les tailles de FlexVol dans ces pilotes. ("Question no 341").
- Ajout de la fonctionnalité d'auto-rétablissement iSCSI pour lancer des analyses SCSI par ID de LUN exact

si des igroups obsolètes sont en cours d'utilisation ("Question no 883").

- Prise en charge supplémentaire des opérations de clonage de volume et de redimensionnement même lorsque le back-end est en mode suspendu.
- Ajout de la possibilité de propager les paramètres de journal configurés par l'utilisateur pour le contrôleur Trident aux pods de nœud Trident.
- Ajout de la prise en charge dans Trident pour utiliser REST par défaut au lieu de ZAPI pour ONTAP versions 9.15.1 et ultérieures.
- Prise en charge des noms de volumes et des métadonnées personnalisés sur les systèmes back-end de stockage ONTAP pour les nouveaux volumes persistants.
- Amélioration du `azure-netapp-files` pilote (ANF) pour activer automatiquement le répertoire de snapshots par défaut lorsque les options de montage NFS sont définies pour utiliser la version 4.x.
- Ajout de la prise en charge de BottlerRocket pour les volumes NFS.
- Ajout de la prise en charge des aperçus techniques de Google Cloud NetApp volumes.

Kubernetes

- Prise en charge de Kubernetes 1.30.
- Ajout de la capacité de Trident DemonSet à nettoyer les montages zombies et les fichiers de suivi résiduels au démarrage ("Question no 883").
- Ajout d'une annotation PVC `trident.netapp.io/luksEncryption` pour l'importation dynamique de volumes LUKS ("Question no 849").
- Prise en compte de la topologie du pilote ANF.
- Ajout de la prise en charge des nœuds Windows Server 2022.

Correctifs

- Correction des défaillances d'installation de Trident suite à des transactions obsolètes.
- Correction de tridentctl pour ignorer les messages d'avertissement de Kubernetes ("Question no 892").
- La priorité du contrôleur Trident a été modifiée `SecurityContextConstraint` en `0` ("Question no 887").
- Les pilotes ONTAP acceptent désormais des volumes inférieurs à 20 Mio ("Problème[#885").
- Correction de Trident pour empêcher la diminution des volumes FlexVol lors de l'opération de redimensionnement pour le pilote ONTAP-SAN.
- Correction de la défaillance d'importation du volume ANF avec NFS v4.1.

Dérations

- Suppression de la prise en charge de EOL Windows Server 2019.

Changements en 24.02

Améliorations

- Prise en charge supplémentaire de Cloud Identity.
  - AKS avec ANF : Azure Workload Identity sera utilisé comme identité cloud.
  - EKS avec FSxN : le rôle IAM AWS sera utilisé comme identité cloud.

- Ajout de la prise en charge de l'installation de Trident en tant que module complémentaire sur le cluster EKS à partir de la console EKS.
- Ajout de la possibilité de configurer et de désactiver l'auto-rétablissement iSCSI ("Question no 864").
- Ajout de la personnalité FSX aux pilotes ONTAP pour permettre l'intégration avec AWS IAM et SecretsManager et pour permettre à Trident de supprimer des volumes FSX avec des sauvegardes ("Question no 453").

Kubernetes

- Prise en charge de Kubernetes 1.29.

Correctifs

- Correction des messages d'avertissement ACP lorsque ACP n'est pas activé ("Question no 866").
- Ajout d'un délai de 10 secondes avant d'effectuer une répartition des clones lors de la suppression d'un snapshot pour les pilotes ONTAP, lorsqu'un clone est associé au snapshot.

Dérations

- Suppression de l'infrastructure d'attepostes in-to des manifestes d'images multi-plates-formes.

Changements en 23.10

Correctifs

- Extension de volume fixe si la nouvelle taille demandée est inférieure à la taille totale des volumes pour les pilotes de stockage ontap-nas et ontap-nas-flexgroup ("Question no 834").
- Taille de volume fixe pour afficher uniquement la taille utilisable du volume lors de l'importation pour les pilotes de stockage ontap-nas et ontap-nas-flexgroup ("Question no 722").
- Conversion de noms FlexVol fixes pour ONTAP-NAS-Economy.
- Correction du problème d'initialisation Trident sur un nœud Windows lors du redémarrage du nœud.

Améliorations

Kubernetes

Prise en charge de Kubernetes 1.28.

Trident

- Ajout de la prise en charge de l'utilisation d'ami (Azure Managed identités) avec le pilote de stockage Azure-netapp-Files.
- Ajout de la prise en charge de NVMe over TCP pour le pilote ONTAP-SAN
- Ajout de la possibilité de suspendre le provisionnement d'un volume lorsque le back-end est défini sur suspendu par l'utilisateur ("Question no 558").

Changements en 23.07.1

Kubernetes: Suppression fixe du démonset pour prendre en charge les mises à niveau sans temps d'arrêt ("Question no 740").

Changements en 23.07

Correctifs

Kubernetes

- Correction de la mise à niveau de Trident pour ignorer les anciens pods bloqués en état de terminaison ("Question no 740").
- Ajout d'une tolérance à la définition de « passagent-trident-version-pod » ("Question no 795").

Trident

- Correction des demandes ZAPI ONTAP pour s'assurer que les numéros de série des LUN sont interrogés lors de l'obtention des attributs de LUN pour identifier et corriger les périphériques iSCSI fantômes pendant les opérations de stadification des nœuds.
- Correction de la gestion des erreurs dans le code du pilote de stockage ("Question no 816").
- Redimensionnement des quotas fixes lors de l'utilisation de pilotes ONTAP avec use-REST=true.
- Création de clones LUN fixes dans ontap-san-Economy.
- Annuler la publication du champ d'informations de `rawDevicePath` à `devicePath`; logique ajoutée pour remplir et récupérer (dans certains cas) `devicePath` légale.

Améliorations

Kubernetes

- Prise en charge supplémentaire de l'importation de snapshots préprovisionnés.
- Déploiement réduit et autorisations linux diaboconfigurées ("Question no 817").

Trident

- Ne rapporte plus le champ d'état pour les volumes et les snapshots « en ligne ».
- Met à jour l'état du back-end si le back-end ONTAP est hors ligne ("Questions #801", "#543").
- Le numéro de série de la LUN est toujours récupéré et publié au cours du workflow ControllerVolumePublish.
- Ajout d'une logique supplémentaire pour vérifier le numéro de série et la taille du périphérique iSCSI à chemins d'accès multiples.
- Vérification supplémentaire des volumes iSCSI pour s'assurer que le périphérique multiacheminement correct n'est pas mis en place.

Amélioration expérimentale

Ajout de la prise en charge de la présentation technique de NVMe over TCP pour le pilote ONTAP-SAN.

Documentation

De nombreuses améliorations de l'organisation et du formatage ont été apportées.

Dérations

Kubernetes

- Suppression de la prise en charge des snapshots v1beta1.
- Suppression de la prise en charge des volumes et des classes de stockage pré-CSI.
- Mise à jour de la version 1.22 de Kubernetes minimale prise en charge.

Changements en 23.04

> Forcer le détachement de volume pour les volumes ONTAP-SAN-* est uniquement pris en charge avec les versions Kubernetes avec le volet fonctionnalité de fermeture de nœud non gracieuse activé. Le détachement forcé doit être activé au moment de l'installation à l'aide du `--enable-force-detach` Indicateur du programme d'installation Trident.

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 de l'opérateur Trident pour qu'elles soient synchronisées avec les autorisations du bundle ("Question no 799").
- 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.
- Résolution du problème où le contrôleur Trident n'a pas pu s'arrêter immédiatement ("Question no 811").
- Correctif ajouté pour afficher la liste de tous les noms de groupes initiateur associés à une LUN spécifiée provisionnée avec des pilotes ontap-san-*.
- Ajout d'un correctif pour permettre l'exécution des processus externes.
- Erreur de compilation corrigée pour l'architecture s390 ("Question no 537").
- Correction d'un niveau de journalisation incorrect lors des opérations de montage de volume ("Question no 781").
- Correction de l'erreur d'assertion de type de potentiel ("Question no 802").

Améliorations

- Kubernetes :
  - Prise en charge de Kubernetes 1.27.
  - Ajout de la prise en charge de l'importation de volumes LUKS.
  - Ajout de la prise en charge du mode d'accès PVC ReadWriteOncePod.
  - Ajout de la prise en charge du détachement forcé pour les volumes ONTAP-SAN-* lors des scénarios d'arrêt de nœud non gracieuse.
  - Tous les volumes ONTAP-SAN-* utiliseront désormais les groupes initiateurs par nœud. Les LUN ne seront mappées qu'aux igroups dont la publication est active sur ces nœuds afin d'améliorer notre niveau de sécurité. Les volumes existants seront basculés de manière opportuniste vers le nouveau schéma d'igroup lorsque Trident détermine qu'il est possible de le faire sans incidence sur les workloads actifs ("Question no 758").

    - Amélioration de la sécurité de Trident en nettoyant les groupes initiateurs gérés par Trident non utilisés à partir de systèmes back-end ONTAP-SAN-*.
- Ajout de la prise en charge des volumes SMB avec Amazon FSX aux pilotes de stockage ontap-nas-Economy et ontap-nas-flexgroup.
- Ajout de la prise en charge des partages SMB avec les pilotes de stockage ontap-nas, ontap-nas-Economy et ontap-nas-flexgroup.
- Ajout de la prise en charge des nœuds arm64 ("Question no 732").
- La procédure d'arrêt de Trident a été améliorée en désactivant d'abord les serveurs d'API ("Question no 811").
- Ajout de la prise en charge de la construction multi plate-forme pour les hôtes Windows et arm64 à Makefile ; voir BUILD.md.

Dérations

Kubernetes: les igroups Backend-scoped ne seront plus créés lors de la configuration de pilotes ontap-san et ontap-san-Economy ("Question no 758").

Changements en 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 1.27 est désormais pris en charge dans Trident. Veuillez mettre à niveau 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.

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 critiques suivantes avant de passer à Trident 22.10.

** informations sur Trident 22,10 **

- Kubernetes 1.25 est désormais pris en charge par Trident. Vous devez effectuer une mise à niveau de Trident vers la version 22.10 avant de passer à Kubernetes 1.25.
- Trident applique désormais strictement l'utilisation de la configuration de chemins d'accès multiples dans les environnements SAN, avec une 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.

- Suppression des tentatives granulaires dans le flux de travail Trident iSCSI pour échouer rapidement 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 effectuer une mise à niveau de Trident vers la version 22.10 avant de passer à 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 fonctions réseau au pilote Azure NetApp Files. ("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"section (PSS) détaillant les autorisations activées par Trident lors de l'installation a été ajoutée.

Changements en 22.04

NetApp améliore et améliore continuellement ses produits et services. Voici quelques-unes des dernières fonctionnalités de Trident. Pour les versions précédentes, reportez-vous à "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")
- Résolution du problème d'échec du clonage d'un volume avec plusieurs règles d'exportation par le pilote Azure NetApp Files.

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 le ralentissement des performances Trident ralentit lors de la création de plus de 32 snapshots sur un volume SolidFire.
- SHA-1 remplacé par SHA-256 lors de la création du certificat SSL.
- Correction du pilote Azure NetApp Files pour permettre la duplication des noms de ressources et limiter les opérations à un seul emplacement.
- Correction du pilote Azure NetApp Files pour permettre la duplication des noms de ressources et limiter les opérations à un seul emplacement.

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 Azure NetApp Files. ("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 dans lequel Trident a consigné une erreur fatale à 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).
- Résolution du problème lié à l'identification par tous les programmes d'installation de Trident de type Kubernetes standard en tant qu'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.

- Lors de la mise à niveau d'un cluster Kubernetes de la version 1.24 vers la version 1.25 ou ultérieure sur lequel Trident est installé, vous devez mettre à jour values.yaml pour définir `excludePodSecurityPolicy` sur `true` ou ajouter la `--set excludePodSecurityPolicy=true helm upgrade` commande avant de pouvoir mettre à niveau le cluster.
- Trident applique maintenant un espace vide `fsType` (`fsType=""`) pour les volumes qui n'ont pas `fsType` spécifié dans leur classe de stockage. Avec Kubernetes 1.17 ou version ultérieure, Trident prend en charge la fourniture d'un espace 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 à l'aide d'un `fsGroup` contexte de sécurité.
- Lors de l'utilisation d'un système back-end sur plusieurs instances Trident, chaque fichier de configuration back-end doit avoir `storagePrefix` une valeur différente pour les systèmes ONTAP back-end ou être utilisé différemment `TenantName` pour les systèmes SolidFire back-end. Trident ne peut pas détecter les volumes créés par d'autres instances de Trident. La tentative de création d'un volume existant sur un système ONTAP ou SolidFire back-end réussit, car Trident considère la création de volume comme une opération puissante. Si `storagePrefix` ou `TenantName` ne diffèrent pas, il peut y avoir des collisions de nom pour les volumes créés sur le même backend.
- Lorsque vous installez Trident (à l'aide de `tridentctl` ou de l'opérateur Trident) et que vous utilisez `tridentctl` pour gérer Trident, vous devez vous assurer que la `KUBECONFIG` variable d'environnement est définie. Ceci est nécessaire pour indiquer le cluster Kubernetes sur lequel `tridentctl` doit fonctionner. Lorsque vous travaillez avec plusieurs environnements Kubernetes, vous devez vous assurer que le `KUBECONFIG` fichier provient correctement.

- 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 avez plusieurs instances de Trident par cluster Kubernetes, Trident ne peut pas communiquer avec d'autres instances et ne peut pas découvrir d'autres volumes qu'elles ont créés. Ce qui entraîne un comportement inattendu et incorrect si plusieurs instances s'exécutent dans un cluster. Il ne doit y avoir qu'une seule instance de Trident par cluster Kubernetes.
- Si des objets basés sur Trident `StorageClass` sont supprimés de Kubernetes alors que Trident est hors ligne, Trident ne supprime pas les classes de stockage correspondantes de sa base de données lorsqu'il est de nouveau en ligne. Vous devez supprimer ces classes de stockage à l'aide de `tridentctl` ou de l'API REST.
- Si un utilisateur supprime un volume persistant provisionné par Trident avant la suppression de la demande de volume persistant correspondante, 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 Trident sur IPv6, vous devez spécifier `managementLIF` et `dataLIF` dans la définition du backend entre crochets. Par exemple `[fd20:8b1e:b258:2000:f816:3eff:feec:0],.`

> Vous ne pouvez pas spécifier `dataLIF` sur un système SAN backend ONTAP. Trident détecte toutes les LIFs 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

- "GitHub pour Trident"
- "Blogs Trident"

Versions antérieures de la documentation

Si vous n'exécutez pas Trident 24.10, la documentation des versions précédentes est disponible sur la base de "Cycle de vie du support Trident".

- "Trident 24.06"
- "Trident 24.02"
- "Trident 23.10"
- "Trident 23.07"
- "Trident 23.04"
- "Trident 23.01"
- "Trident 22.10"
- "Trident 22.07"

- "Trident 22.04"
- "Trident 22.01"

Commencez

Découvrez Trident

Découvrez Trident

Trident est un projet open source entièrement pris en charge et géré par NetApp. Il a été conçu pour vous aider à répondre aux exigences de persistance de vos applications conteneurisées en utilisant des interfaces standard telles que l'interface CSI (Container Storage interface).

Qu'est-ce que Trident ?

NetApp Trident permet de consommer et de gérer les ressources de stockage sur toutes les plateformes de stockage NetApp les plus populaires, dans le cloud public ou sur site, 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.

Trident est un orchestrateur de stockage dynamique conforme à la norme Container Storage interface (CSI) qui s'intègre de manière native à "Kubernetes". Trident s'exécute comme un seul pod de contrôleur et un pod de nœuds sur chaque nœud worker du cluster. Voir "Architecture Trident" pour plus de détails.

Trident propose également une intégration directe avec l'écosystème Docker pour les plateformes de stockage NetApp. Le plug-in de volume Docker (nDVP) de NetApp prend en charge le provisionnement et la gestion des ressources de stockage depuis la plateforme de stockage jusqu'aux hôtes Docker. Voir "Déployez Trident pour Docker" pour plus de détails.

> Si vous utilisez Kubernetes pour la première fois, familiarisez-vous avec le "Concepts et outils Kubernetes".

Testez Trident

Pour tester un disque, demandez l'accès au « déploiement et clonage simplifiés du stockage persistant pour les workloads conteneurisés » "Test Drive NetApp" à l'aide d'une image de laboratoire prête à l'emploi. Le lecteur de test fournit un environnement sandbox avec un cluster Kubernetes à trois nœuds et Trident installés et configurés. C'est un excellent moyen de vous familiariser avec Trident et d'explorer ses fonctionnalités.

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

> N'utilisez pas un cluster Kubernetes que vous créez en utilisant ces instructions dans un environnement de production. Utilisez les guides de déploiement de production fournis par votre distributeur pour les clusters prêts à la production.

Intégration de Kubernetes avec les produits NetApp

Le portefeuille de produits de stockage NetApp s'intègre à de nombreux aspects des clusters Kubernetes avec des fonctionnalités avancées de gestion des données qui améliorent la fonctionnalité, la capacité, les performances et la disponibilité du déploiement Kubernetes.

Amazon FSX pour NetApp ONTAP

"Amazon FSX pour NetApp ONTAP" Est un service AWS entièrement géré qui vous permet de lancer et d'exécuter des systèmes de fichiers optimisés par le système d'exploitation du stockage NetApp ONTAP.

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 ONTAP

"Cloud Volumes ONTAP" Est une appliance de stockage exclusivement logicielle qui exécute le logiciel de gestion des données ONTAP dans le cloud.

Google Cloud NetApp volumes

"Google Cloud NetApp volumes" Est un service de stockage de fichiers entièrement géré dans Google Cloud qui fournit un stockage de fichiers haute performance de grande qualité.

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.

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.

Trident peut provisionner et gérer des terminaux de stockage pour les applications conteneurisées directement sur la plateforme de stockage NetApp HCI sous-jacente.

NetApp ONTAP

"NetApp ONTAP" Il s'agit du système d'exploitation de stockage unifié multiprotocole 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 ces modèles de déploiement ONTAP.

Architecture Trident

Trident s'exécute comme un seul pod de contrôleur et un pod de nœuds sur chaque nœud worker du cluster. Le pod de nœud doit s'exécuter sur n'importe quel hôte sur lequel vous souhaitez potentiellement monter un volume Trident.

Présentation des pods de contrôleur et des nœuds

Trident se déploie comme un seul Pod du contrôleur Trident et un ou plusieurs Pods de nœuds Trident dans le cluster Kubernetes et utilise des conteneurs sidecar Kubernetes standard *CSI* pour simplifier le déploiement des plug-ins CSI. "Conteneurs Sidecar Kubernetes CSI" Sont gérés par la communauté Kubernetes Storage.

Kubernetes "sélecteurs de nœuds" et "tolérances et rejets" sont utilisés pour contraindre un pod à s'exécuter sur un nœud spécifique ou préféré. Vous pouvez configurer des sélecteurs de nœuds et des tolérances pour les pods de contrôleur et de nœud lors de l'installation de Trident.

- 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.

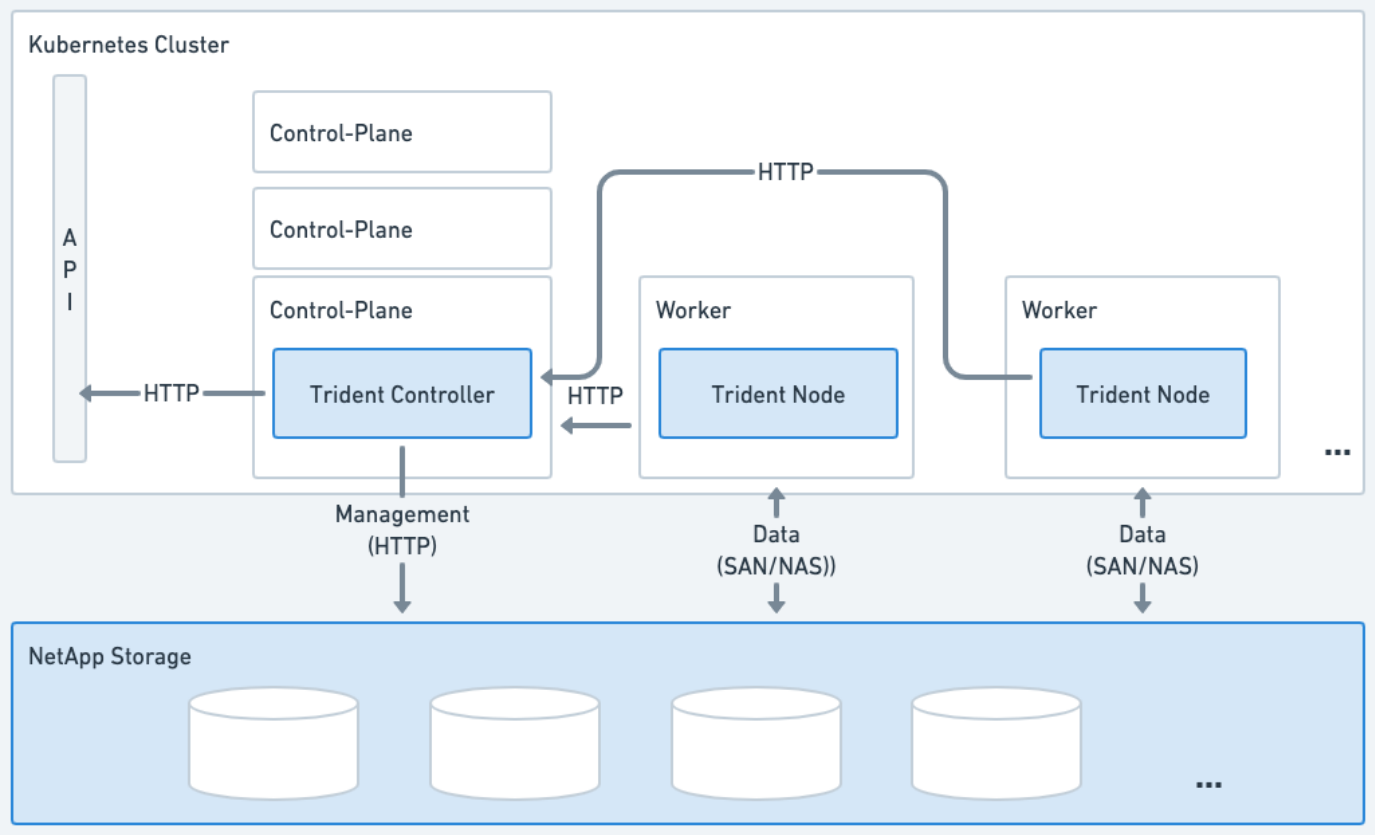


Figure 1. Trident a été déployé sur le cluster Kubernetes

Pod du contrôleur Trident

Le pod du contrôleur Trident est un pod unique exécutant le plug-in du contrôleur CSI.

- Responsable du provisionnement et de la gestion des volumes dans le stockage NetApp
- Géré par un déploiement Kubernetes

- Peut s'exécuter sur le plan de contrôle ou les nœuds workers, selon les paramètres d'installation.

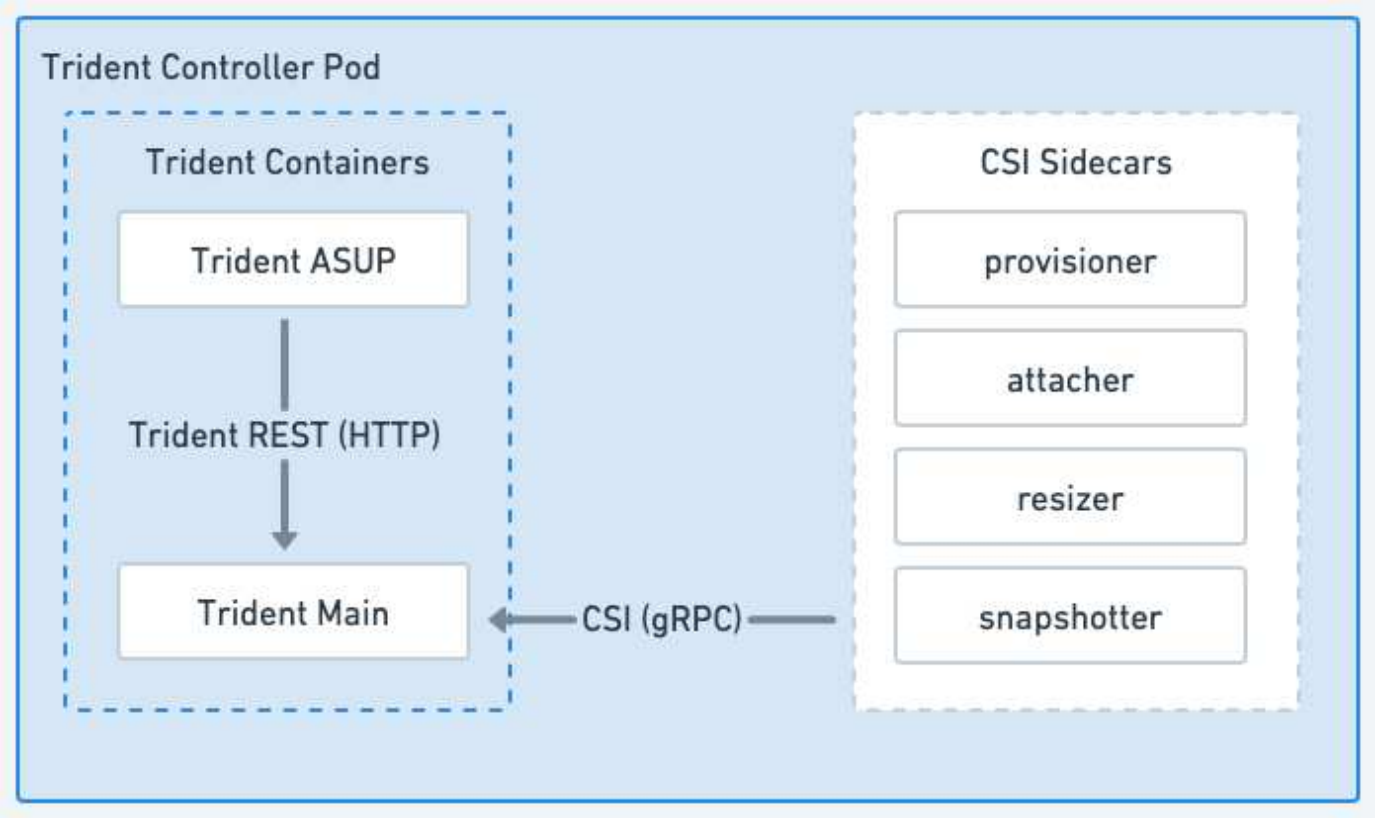


Figure 2. Diagramme du module de contrôleur Trident

Pods de nœuds Trident

Les pods de nœud Trident sont des pods privilégiés exécutant le plug-in CSI Node.

- Responsable du montage et du démontage du stockage des pods qui s'exécutent sur l'hôte
- Géré par un jeu de démonstration Kubernetes
- Doit s'exécuter sur n'importe quel nœud qui montera le stockage NetApp

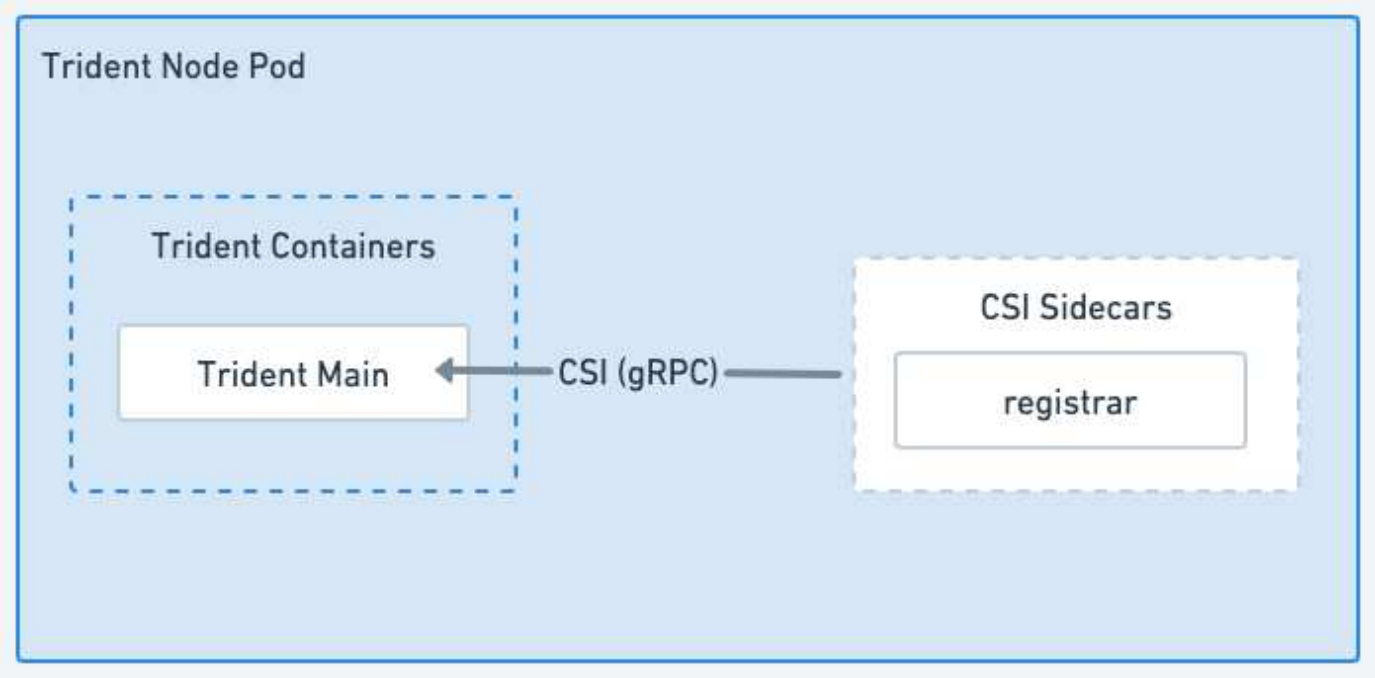


Figure 3. Diagramme Trident Node Pod

Architectures de cluster Kubernetes prises en charge

Trident est pris en charge dans 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.

Concepts

Provisionnement

Le provisionnement dans Trident comporte 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 et nécessite le choix d'un pool de stockage parmi 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 dépend à la fois des attributs demandés de la classe de stockage et des `storagePools` listes , `additionalStoragePools` et `excludeStoragePools` . 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, Trident ajoute ce pool de stockage à l'ensemble des pools de stockage appropriés pour cette classe de stockage. De plus, Trident ajoute à cet ensemble tous les pools de stockage répertoriés dans `additionalStoragePools` la liste, même si leurs attributs ne remplissent pas la totalité ou l'un des attributs demandés de la classe de stockage. Vous devez utiliser cette `excludeStoragePools` liste pour remplacer et supprimer les pools de stockage qui ne sont pas utilisés pour une classe de stockage. Trident effectue un processus similaire chaque fois que vous ajoutez un nouveau système back-end, en vérifiant si ses pools de stockage satisfont à ceux des classes de stockage existantes et en supprimant les éléments marqués comme exclus.

Création du volume

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, Trident obtient d'abord l'ensemble des pools de stockage pour la classe de stockage de ce volume. Si vous spécifiez un protocole pour le volume, Trident supprime les pools de stockage qui ne peuvent pas fournir le protocole demandé (par exemple, un système back-end NetApp HCI/SolidFire ne peut pas fournir de volume basé sur des fichiers alors qu'un système back-end ONTAP ne peut pas fournir de volume basé sur des blocs). Trident répartit de manière aléatoire l'ordre de ce jeu de résultats, pour faciliter une distribution homogène des volumes, puis effectue des itérations via celui-ci, en essayant de provisionner le volume sur chaque pool de stockage à son tour. S'il réussit sur un, il retourne avec succès, en enregistrant les échecs rencontrés dans le processus. Trident renvoie une défaillance **uniquement si** il ne parvient pas à provisionner sur **tous** les pools de stockage disponibles pour la classe et le protocole de stockage demandés.

Snapshots de volume

Découvrez comment Trident gère la création de snapshots de volumes 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 davantage de stabilité, d'évolutivité, de capacité de restauration et de performances que les 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 davantage de stabilité, d'évolutivité, de capacité de restauration et de performances que les 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. Grâce à la technologie FlexClone de ONTAP, il est possible de créer quasi instantanément des copies des jeux de données les plus volumineux, 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 les `ontap-nas` pilotes et `ontap-san`, les snapshots ONTAP sont des copies instantanées 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. Lorsqu'un volume Snapshot créé via Kubernetes est supprimé, Trident supprime le snapshot ONTAP associé. Les snapshots ONTAP qui n'ont pas été créés par Kubernetes peuvent également être supprimés.

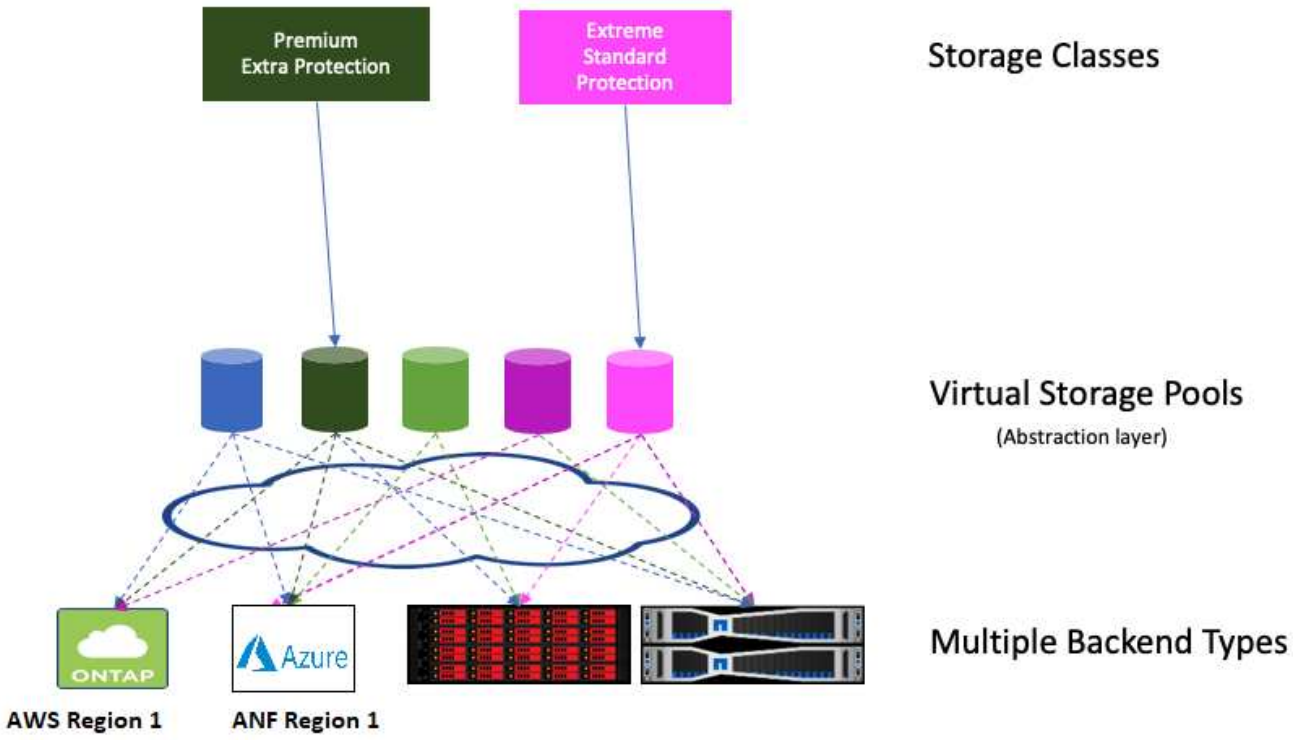
Avec Trident, vous pouvez utiliser Volumesnapshots pour créer de nouveaux volumes persistants à partir de ces volumes. La création de volumes persistants est effectuée à partir de ces copies Snapshot à l'aide de la technologie FlexClone pour les systèmes back-end ONTAP et CVS pris en charge. Lors de 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` pilote utilise des clones de volume du logiciel 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 Trident et Kubernetes `StorageClasses`. Ils permettent à un administrateur de définir des aspects, tels que l'emplacement, les performances et la protection de chaque back-end de manière commune et indépendante du back-end, sans `StorageClass` spécifier le backend physique, le pool back-end ou le type de 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 l'un des systèmes Trident back-end 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

En savoir plus sur l'utilisation de 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 Trident en mode CSI, vous pouvez ignorer cette section. 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

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

Trident associe de nouveaux volumes aux groupes d'accès configurés, mais ne crée pas et ne gère pas eux-mêmes les groupes d'accès. Les groupes d'accès doivent exister avant l'ajout du système back-end de stockage à Trident et doivent contenir les IQN iSCSI de chaque nœud du cluster Kubernetes pouvant

potentiellement monter les volumes provisionnés par ce 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 relatives aux groupes d'accès aux volumes, reportez-vous à la section "ici".

Si vous modifiez la configuration à partir d'une configuration utilisant la configuration 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.

Démarrage rapide pour Trident

Vous pouvez installer Trident et commencer à gérer les ressources de stockage en quelques étapes. Avant de commencer, consultez "Configuration requise pour Trident".

Pour Docker, reportez-vous "Trident pour Docker"à la .

1 Installez Trident

Trident propose plusieurs méthodes et modes d'installation optimisés pour un large éventail d'environnements et d'organisations.

"Installation de Trident"

2 Préparez le nœud de travail

Tous les nœuds workers du cluster Kubernetes doivent pouvoir monter les volumes provisionnés pour vos pods.

"Préparez le nœud de travail"

3 Créer un back-end

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

"Configurer un back-end" de votre système de stockage

4 Créez une classe de stockage Kubernetes

L'objet StorageClass Kubernetes spécifie Trident comme provisionneur et permet de créer une classe de stockage pour provisionner des volumes avec des attributs personnalisables. Trident crée une classe de stockage correspondante pour les objets Kubernetes qui spécifient le mécanisme de provisionnement Trident.

"Créer une classe de stockage"

5 **Provisionner un volume**

Un *PersistentVolume* (PV) est une ressource de stockage physique provisionnée par l'administrateur du cluster sur un cluster Kubernetes. La demande de volume persistant *PersistentVolumeClaim* (PVC) est une demande d'accès au volume persistant sur le cluster.

Créez un volume persistant et une demande de volume persistant qui utilisent la classe de stockage Kubernetes configurée pour demander l'accès au volume persistant. Vous pouvez ensuite monter le volume persistant sur un pod.

"Provisionner un volume"

Et la suite ?

Vous pouvez à présent ajouter des systèmes back-end supplémentaires, gérer les classes de stockage, gérer les systèmes back-end et effectuer des opérations de volume.

De formation

Avant d'installer Trident, vous devez vérifier la configuration système requise. Il se peut que les systèmes back-end spécifiques présentent des exigences supplémentaires.

Informations critiques sur Trident

Vous devez lire les informations critiques suivantes sur Trident.

** informations sur le Trident **

- Kubernetes 1.31 est désormais pris en charge dans Trident. Mise à niveau de Trident avant la mise à niveau de Kubernetes.
- Trident applique strictement l'utilisation de la configuration de chemins d'accès multiples dans les environnements SAN, avec une 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

Trident prend en charge plusieurs moteurs de mise en conteneurs et orchestrateurs, notamment :

- Anthos sur site (VMware) et Anthos sur bare Metal 1.16
- Kubernetes 1.25 - 1.31
- OpenShift 4.10 - 4.17

- Rancher Kubernetes Engine 2 (RKE2) v1.28.5+rke2r1

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

- Anthos sur site (VMware) et Anthos sur bare Metal 1.16
- Kubernetes 1.25 - 1.31
- OpenShift 4.10 - 4.17
- Rancher Kubernetes Engine 2 (RKE2) v1.28.5+rke2r1

Trident fonctionne également avec de nombreuses autres offres Kubernetes entièrement gérées et autogérées, notamment Google Kubernetes Engine (GKE), Amazon Elastic Kubernetes Services (EKS), Azure Kubernetes Service (AKS), Mirantis Kubernetes Engine (MKE) et le portefeuille VMware Tanzu.

Trident et ONTAP peuvent être utilisés comme fournisseur de stockage pour "KubeVirt".

> Avant de mettre à niveau un cluster Kubernetes de la version 1.24 vers la version 1.25 ou ultérieure sur "Mettre à niveau une installation Helm"lequelTrident est installé, reportez-vous à la section .

Systèmes back-end pris en charge (stockage)

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

- Amazon FSX pour NetApp ONTAP
- Azure NetApp Files
- Cloud Volumes ONTAP
- Google Cloud NetApp volumes
- 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'Trident et les versions de Kubernetes qu'il prend en charge.

Fonction	Version Kubernetes	Portes-fonctions requises ?
Trident	1,25 - 1,31	Non
Snapshots de volume	1,25 - 1,31	Non
Volume persistant à partir des copies Snapshot des volumes	1,25 - 1,31	Non
Redimensionnement PV iSCSI	1,25 - 1,31	Non
Chap bidirectionnel ONTAP	1,25 - 1,31	Non

Fonction	Version Kubernetes	Portes-fonctions requises ?
Règles d'exportation dynamiques	1,25 - 1,31	Non
Opérateur Trident	1,25 - 1,31	Non
Topologie CSI	1,25 - 1,31	Non

Systèmes d'exploitation hôtes testés

Bien que Trident ne prenne pas officiellement en charge des systèmes d'exploitation spécifiques, les éléments suivants sont connus pour fonctionner :

- Versions de Red Hat CoreOS (RHCOS) prises en charge par OpenShift Container Platform (AMD64 et ARM64)
- RHEL 8+ (AMD64 ET ARM64)

  NVMe/TCP requiert RHEL 9 ou version ultérieure.

- Ubuntu 22.04 ou version ultérieure (AMD64 et ARM64)
- Windows Server 2022

Par défaut, Trident s'exécute dans un conteneur et s'exécute donc sur n'importe quel travailleur Linux. Toutefois, ces derniers doivent pouvoir monter les volumes offerts par Trident à l'aide du client NFS standard ou de l'initiateur iSCSI, en fonction des systèmes back-end que vous utilisez.

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 worker, vous devez installer les outils NFS, iSCSI ou NVMe en fonction de votre sélection de pilotes.

"Préparez le nœud de travail"

Configuration du système de stockage

Trident peut nécessiter des modifications d'un système de stockage avant qu'une configuration back-end ne puisse l'utiliser.

"Configuration des systèmes back-end"

Ports Trident

Trident requiert l'accès à des ports spécifiques pour la communication.

"Ports Trident"

Images de conteneur et versions Kubernetes correspondantes

Pour les installations à air comprimé, la liste suivante est une référence aux images de conteneur nécessaires à l'installation de Trident. Utiliser `tridentctl images` la commande pour vérifier la liste des images de conteneur nécessaires.

Versions de Kubernetes	Image de conteneur
v1.25.0, v1.26.0, v1.27.0, v1.28.0, v1.29.0, v1.30.0, v1.31.0	• docker.io/netapp/trident : 24.10.0 • docker.io/netapp/trident-autosupport:24.10 • registry.k8s.io/sig-storage/csi-provisionneur:v5.1.0 • registry.k8s.io/sig-storage/csi-attacher:v4.7.0 • registry.k8s.io/sig-storage/csi-resizer:v1.12.0 • registry.k8s.io/sig-storage/csi-snapshotter:v8.1.0 • registry.k8s.io/sig-storage/csi-node-driver-registratr:v2.12.0 • docker.io/netapp/trident-operator:24.10.0 (en option)

Installation de Trident

En savoir plus sur l'installation de Trident

Pour garantir que Trident peut être installé dans un large éventail d'environnements et d'organisations, NetApp propose de nombreuses options d'installation. Vous pouvez installer Trident à l'aide de l'opérateur Trident (manuellement ou à l'aide de Helm) ou de `tridentctl`. Cette rubrique fournit des informations importantes pour sélectionner le processus d'installation qui vous convient.

Informations critiques sur Trident 24.06

Vous devez lire les informations critiques suivantes sur Trident.

** informations sur le Trident **

- Kubernetes 1.31 est désormais pris en charge dans Trident. Mise à niveau de Trident avant la mise à niveau de Kubernetes.
- Trident applique strictement l'utilisation de la configuration de chemins d'accès multiples dans les environnements SAN, avec une 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

Qu'il s'agisse d'un déploiement manuel ou à l'aide d'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" utiliser les attributs dans la `TridentOrchestrator` ressource personnalisée (CR).

L'utilisateur de Trident présente les avantages suivants :

**Objet Trident **

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

** compte : « »**

L'opérateur cluster-scoped Trident gère les ressources associées à une installation Trident au niveau du cluster. Cela réduit les erreurs pouvant être provoquées lors de la maintenance des ressources du cluster-scoped à l'aide d'un opérateur namespace-scoped. Ceci est essentiel pour l'auto-rétablissement et l'application de correctifs.

** - Capcuratif de la prise **

L'opérateur surveille l'installation de Trident et prend activement des mesures pour résoudre les problèmes, tels que la suppression du déploiement ou la modification accidentelle. Un `trident-operator-<generated-id>` pod est créé pour associer une `TridentOrchestrator` demande de modification à une installation Trident. Cela permet de s'assurer qu'il n'y a qu'une seule instance de Trident dans le cluster et de contrôler sa configuration, en s'assurant que l'installation est idempuissant. 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.

Par exemple, prenons un scénario dans lequel vous devez activer Trident pour générer des journaux de débogage. Pour ce faire, mettez votre `TridentOrchestrator` à `spec.debug 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.

** **

L'opérateur Trident dont le périmètre est défini dans le cluster permet la suppression complète des ressources dont le périmètre est défini dans le cluster. Les utilisateurs peuvent désinstaller complètement Trident et réinstaller facilement.

** : 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 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 de Trident. Si Trident a déjà été installé avec l'opérateur, un avertissement s'affiche pour indiquer que Trident est installé sur une version Kubernetes non prise en charge.

À l'aide de `tridentctl`

Si vous disposez d'un déploiement existant qui doit être mis à niveau ou si vous cherchez à personnaliser votre déploiement, vous devriez envisager . Il s'agit là d'une méthode classique de déploiement de Trident.

Vous pouvez générer les manifestes pour les ressources Trident. Cela inclut le déploiement, le démonset, le compte de service et le rôle de cluster que Trident crée dans le cadre de son installation.

> ⓘ À partir de la version 22.04, les clés AES ne seront plus régénérées à chaque installation de Trident. Avec cette version, Trident installera un nouvel objet secret qui perdure entre les installations. Cela signifie que `tridentctl` dans 22.04 peut désinstaller les versions précédentes de Trident, mais les versions antérieures ne peuvent pas désinstaller les installations 22.04. 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

C'est le moyen le plus simple d'installer Trident et 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 les (`registry.k8s.io`images Trident (`docker.io`) et CSI) requises.

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

- Extrait les images conteneur sur Internet
- Crée un démonset de déploiement ou de nœud qui fait tourner les pods Trident sur tous les nœuds éligibles dans le 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 la machine distante à partir de laquelle vous souhaitez déployer 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"

Méthode	Mode d'installation
`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 Trident. Si vous avez déployé avec `tridentctl`, vous devez utiliser la version appropriée du `tridentctl` binaire pour désinstaller Trident. De même, si vous déployez avec l'opérateur, vous devez modifier la `TridentOrchestrator` CR et définir `spec.uninstall=true` pour désinstaller Trident.
- Si vous avez un déploiement basé sur l'opérateur que vous souhaitez supprimer et utiliser à la place `tridentctl` pour déployer Trident, vous devez d'abord modifier `TridentOrchestrator` et définir `spec.uninstall=true` sur désinstaller 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 de 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 Trident. Ce processus s'applique aux installations où les images de conteneur requises par 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".

Informations critiques sur Trident 24.10

Vous devez lire les informations critiques suivantes sur Trident.

** informations sur le Trident **

- Kubernetes 1.31 est désormais pris en charge dans Trident. Mise à niveau de Trident avant la mise à niveau de Kubernetes.
- Trident applique strictement l'utilisation de la configuration de chemins d'accès multiples dans les environnements SAN, avec une 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 programme d'installation de Trident contient tout ce dont vous avez besoin pour déployer l'opérateur Trident et installer Trident. Téléchargez et extrayez la dernière version du programme d'installation de Trident sur "La section *Assets* sur GitHub".

```
wget https://github.com/NetApp/trident/releases/download/v24.10.0/trident-
installer-24.10.0.tar.gz
tar -xf trident-installer-24.10.0.tar.gz
cd trident-installer
```

Étape 2 : créez le `TridentOrchestrator` CRD

Créez la `TridentOrchestrator` définition de ressource personnalisée (CRD). Vous créez ensuite des `TridentOrchestrator` ressources personnalisées. 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 de Trident fournit un fichier de bundle qui peut être utilisé pour installer l'opérateur et créer des objets associés. Le fichier de bundle est un moyen facile de déployer l'opérateur et d'installer Trident à l'aide d'une configuration par défaut.

- Pour les clusters exécutant Kubernetes 1.24, utilisez `bundle_pre_1_25.yaml`.
- Pour les clusters exécutant Kubernetes 1.25 ou version ultérieure, utilisez `bundle_post_1_25.yaml`.

Avant de commencer

- Par défaut, le programme d'installation de Trident déploie l'opérateur dans `trident` espace de noms. Si le `trident` l'espace de noms n'existe pas. créez-le à l'aide des éléments suivants :

  ```
  kubectl apply -f deploy/namespace.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`.

  a. Créer le `kustomization.yaml` en utilisant la commande suivante où <*bundle.yaml*> est `bundle_pre_1_25.yaml` ou `bundle_post_1_25.yaml` Basé sur votre version de Kubernetes.

  ```
  cp deploy/kustomization_<bundle.yaml> deploy/kustomization.yaml
  ```

  b. Compilez le bundle à l'aide de la commande suivante où <*bundle.yaml*> est `bundle_pre_1_25.yaml` ou `bundle_post_1_25.yaml` Basé sur votre version de Kubernetes.

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

Étapes

1. Créer les ressources et déployer l'opérateur :

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

2. Vérifiez que l'opérateur, le déploiement et les réplicateurs ont été créés.

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

> 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 `TridentOrchestrator` et installer Trident. Vous pouvez éventuellement "Personnalisez votre installation de Trident"utiliser les attributs de la `TridentOrchestrator` spécification.

```
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
  nodePrep:
  - iscsi
Status:
  Current Installation Params:
    IPv6:                      false
    Autosupport Hostname:
    Autosupport Image:         netapp/trident-autosupport:24.10
    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:24.10.0
  Message:                  Trident installed  Namespace:
trident
  Status:                   Installed
  Version:                  v24.10.0
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 Trident à l'aide de ce `TridentOrchestrator` CR.
Installé	Trident a été installé avec succès.
Désinstallation	L'opérateur désinstalle Trident, car `spec.uninstall=true`.
Désinstallé	Trident est désinstallé.
Échec	L'opérateur n'a pas pu installer, corriger, mettre à jour ou désinstaller Trident ; l'opérateur essaiera automatiquement de récupérer à partir de 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 si l'installation de Trident est terminée en vérifiant l'état 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 de Trident installée.

```
./tridentctl -n trident version

+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 24.10.0        | 24.10.0        |
+----------------+----------------+
```

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

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

Informations critiques sur Trident 24.10

Vous devez lire les informations critiques suivantes sur Trident.

** informations sur le Trident **

- Kubernetes 1.31 est désormais pris en charge dans Trident. Mise à niveau de Trident avant la mise à niveau de Kubernetes.
- Trident applique strictement l'utilisation de la configuration de chemins d'accès multiples dans les environnements SAN, avec une 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 programme d'installation de Trident contient tout ce dont vous avez besoin pour déployer l'opérateur Trident et installer Trident. Téléchargez et extrayez la dernière version du programme d'installation de Trident sur "La section *Assets* sur GitHub".

```
wget https://github.com/NetApp/trident/releases/download/v24.10.0/trident-
installer-24.10.0.tar.gz
tar -xf trident-installer-24.10.0.tar.gz
cd trident-installer
```

Étape 2 : créez le `TridentOrchestrator` CRD

Créez la `TridentOrchestrator` définition de ressource personnalisée (CRD). Vous créez ensuite des `TridentOrchestrator` ressources personnalisées. 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`, mettez à jour `image: docker.io/netapp/trident-operator:24.10.0` 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:24.10.0` si vos images se trouvent toutes dans un registre.

- `image: <your-registry>/netapp/trident-operator:24.10.0` 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 fournit un fichier de bundle qui peut être utilisé pour installer l'opérateur et créer des objets associés. Le fichier de bundle est un moyen facile de déployer l'opérateur et d'installer Trident à l'aide d'une configuration par défaut.

- Pour les clusters exécutant Kubernetes 1.24, utilisez `bundle_pre_1_25.yaml`.
- Pour les clusters exécutant Kubernetes 1.25 ou version ultérieure, utilisez `bundle_post_1_25.yaml`.

Avant de commencer

- Par défaut, le programme d'installation de Trident déploie l'opérateur dans `trident` espace de noms. Si le `trident` l'espace de noms n'existe pas. créez-le à l'aide des éléments suivants :

```
kubectl apply -f deploy/namespace.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`.

  a. Créer le `kustomization.yaml` en utilisant la commande suivante où *<bundle.yaml>* est `bundle_pre_1_25.yaml` ou `bundle_post_1_25.yaml` Basé sur votre version de Kubernetes.

  ```
  cp deploy/kustomization_<bundle.yaml> deploy/kustomization.yaml
  ```

  b. Compilez le bundle à l'aide de la commande suivante où *<bundle.yaml>* est `bundle_pre_1_25.yaml` ou `bundle_post_1_25.yaml` Basé sur votre version de Kubernetes.

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

Étapes

1. Créer les ressources et déployer l'opérateur :

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

2. Vérifiez que l'opérateur, le déploiement et les réplicateurs ont été créés.

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

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:24.10"
tridentImage: "<your-registry>/trident:24.10.0"
```

Images dans différents registres

```
imageRegistry: "<your-registry>"
autosupportImage: "<your-registry>/trident-autosupport:24.10"
tridentImage: "<your-registry>/trident:24.10.0"
```

Étape 6 : créez le `TridentOrchestrator` Et installer Trident

Vous pouvez maintenant créer `TridentOrchestrator` et installer Trident. Vous pouvez également "Personnalisez votre installation de Trident"utiliser les attributs de la `TridentOrchestrator` spécification. 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>/trident-autosupport:24.10
  Debug:              true
  Image Registry:     <your-registry>
  Namespace:          trident
  Trident Image:      <your-registry>/trident:24.10.0
Status:
  Current Installation Params:
    IPv6:                       false
    Autosupport Hostname:
    Autosupport Image:          <your-registry>/trident-autosupport:24.10
    Autosupport Proxy:
    Autosupport Serial Number:
    Debug:                      true
    Http Request Timeout:       90s
    Image Pull Secrets:
    Image Registry:       <your-registry>
    k8sTimeout:           30
    Kubelet Dir:          /var/lib/kubelet
    Log Format:           text
    Probe Port:           17546
    Silence Autosupport:  false
    Trident Image:        <your-registry>/trident:24.10.0
  Message:                Trident installed
  Namespace:              trident
  Status:                 Installed
  Version:                v24.10.0
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 Trident à l'aide de ce `TridentOrchestrator` CR.
Installé	Trident a été installé avec succès.
Désinstallation	L'opérateur désinstalle Trident, car `spec.uninstall=true`.
Désinstallé	Trident est désinstallé.
Échec	L'opérateur n'a pas pu installer, corriger, mettre à jour ou désinstaller Trident ; l'opérateur essaiera automatiquement de récupérer à partir de 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 si l'installation de Trident est terminée en vérifiant l'état 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 de Trident installée.

```
./tridentctl -n trident version

+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 24.10.0        | 24.10.0        |
+----------------+----------------+
```

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

Vous pouvez déployer l'opérateur Trident et installer Trident à l'aide de Helm. Ce processus s'applique aux installations où les images de conteneur requises par 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".

Informations critiques sur Trident 24.10

Vous devez lire les informations critiques suivantes sur Trident.

informations sur le Trident

- Kubernetes 1.31 est désormais pris en charge dans Trident. Mise à niveau de Trident avant la mise à niveau de Kubernetes.
- Trident applique strictement l'utilisation de la configuration de chemins d'accès multiples dans les environnements SAN, avec une 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.

Déployez l'opérateur Trident et installez Trident à l'aide d'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 Trident Helm :

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

2. Utilisez `helm install` et spécifiez un nom pour votre déploiement comme dans l'exemple suivant où `100.2404.0` est la version de Trident que vous installez.

```
helm install <name> netapp-trident/trident-operator --version 100.2410.0
--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 la commande suivante où `100.2410.0` est la version de Trident que vous installez :

```
helm install <name> netapp-trident/trident-operator --version 100.2410.0
--create-namespace --namespace trident --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	

Option	Description	Valeur par défaut
`podAnnotations`	Annotations de pod	
`deploymentAnnotations`	Annotations de déploiement	
`tolerations`	Tolérances pour l'affectation de pod	

`affinity`	Affinité pour l'affectation de pod	`affinity:`
		`  nodeAffinity:`
		
		`requiredDuringSchedulingIgnoredDuringExecution:`
		`  nodeSelectorTerms:`
		`  - matchExpressions:`
		`  - key: kubernetes.io/arch`
		`  operator: In`
		`  values:`
		`  - arm64`
		`  - amd64`
		`  - key: kubernetes.io/os`
		`  operator: In`
		`  values:`
		`  - linux`
		
		Ne supprimez pas l'affinité par défaut du fichier values.yaml. Lorsque vous souhaitez fournir une affinité personnalisée, étendez l'affinité par défaut.

`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.	

Option	Description	Valeur par défaut
`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 pour les `trident-operator` images , `trident` et autres. Laissez vide pour accepter la valeur par défaut. IMPORTANT : lorsque vous installez Trident dans un référentiel privé, si vous utilisez le `imageRegistry` commutateur pour spécifier l'emplacement du référentiel, n'utilisez pas `/netapp/` dans le chemin du référentiel.	""
`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"`
`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.	""

Option	Description	Valeur par défaut
`tridentIPv6`	Permet d'activer Trident pour 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 les rapports AutoSupport périodiques Trident.	`false`
`tridentAutosupportImageTag`	Permet de remplacer la balise de l'image pour le conteneur Trident AutoSupport.	`<version>`
`tridentAutosupportProxy`	Permet au conteneur Trident AutoSupport de téléphoner à domicile via un proxy HTTP.	`""`
`tridentLogFormat`	Définit le format de journalisation Trident (`text`ou `json`).	`"text"`
`tridentDisableAuditLog`	Désactive l'enregistreur d'audit Trident.	`true`
`tridentLogLevel`	Permet de définir le niveau de journal de Trident sur `trace` , `debug`, , `info`, `warn`, `error` ou `fatal`.	`"info"`
`tridentDebug`	Permet de définir le niveau de journal de Trident sur `debug`.	`false`
`tridentLogWorkflows`	Permet d'activer des flux de travail Trident spécifiques pour la consignation des traces ou la suppression des journaux.	`""`
`tridentLogLayers`	Permet d'activer des couches Trident spécifiques pour la consignation des tracés ou la suppression des journaux.	`""`
`tridentImage`	Permet le remplacement complet de l'image pour Trident.	`""`
`tridentImageTag`	Permet de remplacer la balise de l'image pour Trident.	`""`

Option	Description	Valeur par défaut
`tridentProbePort`	Permet de remplacer le port par défaut utilisé pour les sondes de disponibilité/préparation Kubernetes.	`""`
`windows`	Permet d'installer 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`
`cloudProvider`	Réglez sur `"Azure"` Lors de l'utilisation d'identités gérées ou d'une identité de cloud sur un cluster AKS. Défini sur AWS lors de l'utilisation d'une identité de cloud sur un cluster EKS.	`""`
`cloudIdentity`	Défini sur l'identité de la charge de travail (« Azure.Workload.Identity/client-ID: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxxxx ») lors de l'utilisation de l'identité cloud sur un cluster AKS. Défini sur le rôle IAM AWS (« eks.amazonaws.com/role-arn: arn:aws:iam::123456:role/Trident-role ») lors de l'utilisation de l'identité cloud sur un cluster EKS.	`""`
`iscsiSelfHealingInterval`	Intervalle d'appel de l'auto-rétablissement iSCSI.	`5m0s`
`iscsiSelfHealingWaitTime`	Durée après laquelle l'auto-rétablissement iSCSI lance une tentative de résolution d'une session obsolète en effectuant une déconnexion et une connexion ultérieure.	`7m0s`
`nodePrep`	Permet à Trident de préparer les nœuds du cluster Kubernetes à gérer les volumes à l'aide du protocole de stockage de données spécifié. **Actuellement, `iscsi` est la seule valeur prise en charge.**	

Présentation des pods de contrôleur et des nœuds

Trident s'exécute comme un seul pod de contrôleur, plus un pod de nœud sur chaque nœud worker du cluster. Le pod de nœud doit s'exécuter sur n'importe quel hôte sur lequel vous souhaitez potentiellement monter un

volume 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.

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

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

Informations critiques sur Trident 24.10

Vous devez lire les informations critiques suivantes sur Trident.

** informations sur le Trident **

- Kubernetes 1.31 est désormais pris en charge dans Trident. Mise à niveau de Trident avant la mise à niveau de Kubernetes.
- Trident applique strictement l'utilisation de la configuration de chemins d'accès multiples dans les environnements SAN, avec une 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.

Déployez l'opérateur Trident et installez Trident à l'aide d'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".

> Lors de l'installation de Trident dans un référentiel privé, si vous utilisez le `imageRegistry` commutateur pour spécifier l'emplacement du référentiel, n'utilisez pas `/netapp/` dans le chemin du référentiel.

Étapes

1. Ajout du référentiel Trident Helm :

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

2. Utilisez `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, `100.2410.0` est la version de Trident que vous installez.

Images dans un registre

```
helm install <name> netapp-trident/trident-operator --version
100.2410.0 --set imageRegistry=<your-registry> --create-namespace
--namespace <trident-namespace> --set nodePrep={iscsi}
```

Images dans différents registres

```
helm install <name> netapp-trident/trident-operator --version
100.2410.0 --set imageRegistry=<your-registry> --set
operatorImage=<your-registry>/trident-operator:24.10.0 --set
tridentAutosupportImage=<your-registry>/trident-autosupport:24.06
--set tridentImage=<your-registry>/trident:24.10.0 --create
-namespace --namespace <trident-namespace> --set nodePrep={iscsi}
```

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 la commande suivante où `100.2410.0`

est la version de Trident que vous installez :

```
helm install <name> netapp-trident/trident-operator --version 100.2410.0
--create-namespace --namespace trident --set tridentDebug=true
```

Pour ajouter la valeur nodePrep, exécutez la commande suivante :

```
helm install <name> netapp-trident/trident-operator --version 100.2406.0
--create-namespace --namespace trident --set nodePrep={iscsi}
```

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.

> Ne supprimez pas l'affinité par défaut du fichier values.yaml. Lorsque vous souhaitez fournir une affinité personnalisée, étendez l'affinité 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	

Option	Description	Valeur par défaut

`affinity`	Affinité pour l'affectation de pod	`affinity:`
		`  nodeAffinity:`
		
		`requiredDuringSchedul`
		`ingIgnoredDuringExecu`
		`tion:`
		
		`nodeSelectorTerms:`
		`  -`
		`matchExpressions:`
		`  - key:`
		`kubernetes.io/arch`
		
		`operator: In`
		`  values:`
		`  - arm64`
		`  - amd64`
		`  - key:`
		`kubernetes.io/os`
		
		`operator: In`
		`  values:`
		`  - linux`
		
		Ne supprimez pas l'affinité par défaut du fichier values.yaml. Lorsque vous souhaitez fournir une affinité personnalisée, étendez l'affinité par défaut.

`tridentControllerPluginNod` `eSelector`	Sélecteurs de nœuds supplémentaires pour les pods. Reportez-vous à la section "Présentation des pods de contrôleur et des nœuds" pour plus d'informations.	

Option	Description	Valeur par défaut
`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 pour les `trident-operator` images , `trident` et autres. Laissez vide pour accepter la valeur par défaut. IMPORTANT : lorsque vous installez Trident dans un référentiel privé, si vous utilisez le `imageRegistry` commutateur pour spécifier l'emplacement du référentiel, n'utilisez pas `/netapp/` dans le chemin du référentiel.	« »
`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"`
`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`

Option	Description	Valeur par défaut
`operatorImage`	Permet la neutralisation complète de l'image pour `trident-operator`.	« »
`operatorImageTag`	Permet de remplacer la balise du `trident-operator` image.	« »
`tridentIPv6`	Permet d'activer Trident pour 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 les rapports AutoSupport périodiques Trident.	`false`
`tridentAutosupportImageTag`	Permet de remplacer la balise de l'image pour le conteneur Trident AutoSupport.	`<version>`
`tridentAutosupportProxy`	Permet au conteneur Trident AutoSupport de téléphoner à domicile via un proxy HTTP.	« »
`tridentLogFormat`	Définit le format de journalisation Trident (`text`ou `json`).	`"text"`
`tridentDisableAuditLog`	Désactive l'enregistreur d'audit Trident.	`true`
`tridentLogLevel`	Permet de définir le niveau de journal de Trident sur `trace` , `debug`, , `info`, `warn`, `error` ou `fatal`.	`"info"`
`tridentDebug`	Permet de définir le niveau de journal de Trident sur `debug`.	`false`
`tridentLogWorkflows`	Permet d'activer des flux de travail Trident spécifiques pour la consignation des traces ou la suppression des journaux.	« »
`tridentLogLayers`	Permet d'activer des couches Trident spécifiques pour la consignation des tracés ou la suppression des journaux.	« »

Option	Description	Valeur par défaut
`tridentImage`	Permet le remplacement complet de l'image pour Trident.	« »
`tridentImageTag`	Permet de remplacer la balise de l'image pour Trident.	« »
`tridentProbePort`	Permet de remplacer le port par défaut utilisé pour les sondes de disponibilité/préparation Kubernetes.	« »
`windows`	Permet d'installer 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`
`nodePrep`	Permet à Trident de préparer les nœuds du cluster Kubernetes à gérer les volumes à l'aide du protocole de stockage de données spécifié. **Actuellement, `iscsi` est la seule valeur prise en charge.**	

Personnalisez l'installation de l'opérateur Trident

L'opérateur Trident vous permet de personnaliser l'installation Trident à l'aide des attributs de la `TridentOrchestrator` spécification. Si vous souhaitez personnaliser l'installation au-delà des `TridentOrchestrator` arguments autorisés, envisagez d'utiliser `tridentctl` pour générer des manifestes YAML personnalisés à modifier si nécessaire.

Présentation des pods de contrôleur et des nœuds

Trident s'exécute comme un seul pod de contrôleur, plus un pod de nœud sur chaque nœud worker du cluster. Le pod de nœud doit s'exécuter sur n'importe quel hôte sur lequel vous souhaitez potentiellement monter un volume 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 pour indiquer l'espace de noms dans `TridentOrchestrator` lequel Trident est installé. Ce paramètre **ne peut pas être mis à jour après l'installation de Trident**. Si vous essayez de le faire, l' `TridentOrchestrator`état passe à `Failed`. Trident n'est pas destiné à 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 Trident dans	`"default"`
`debug`	Activer le débogage pour Trident	`false`
`enableForceDetach`	`ontap-san`, `ontap-san-economy` et `ontap-nas-economy` uniquement. Fonctionne avec Kubernetes non-Grass Node Shutdown (NGN) pour autoriser les administrateurs du cluster à migrer en toute sécurité les workloads avec des volumes montés vers de nouveaux nœuds en cas de problème.	`false`
`windows`	Réglage sur `true` Active l'installation sur les nœuds de travail Windows.	`false`
`cloudProvider`	Réglez sur `"Azure"` Lors de l'utilisation d'identités gérées ou d'une identité de cloud sur un cluster AKS. Défini sur AWS lors de l'utilisation d'une identité de cloud sur un cluster EKS.	`""`
`cloudIdentity`	Défini sur l'identité de la charge de travail (« Azure.Workload.Identity/client-ID: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxxxx ») lors de l'utilisation de l'identité cloud sur un cluster AKS. Défini sur le rôle IAM AWS (« eks.amazonaws.com/role-arn: arn:aws:iam::123456:role/Trident-role ») lors de l'utilisation de l'identité cloud sur un cluster EKS.	`""`
`IPv6`	Installez Trident sur IPv6	faux
`k8sTimeout`	Délai d'expiration pour les opérations Kubernetes	`30sec`
`silenceAutosupport`	N'envoyez pas de packs AutoSupport à NetApp automatiquement	`false`
`autosupportImage`	Image conteneur pour la télémétrie AutoSupport	`"netapp/trident-autosupport:24.10"`
`autosupportProxy`	Adresse/port d'un proxy pour l'envoi de AutoSupport Télémétrie	`"http://proxy.example.com:8888"`
`uninstall`	Indicateur utilisé pour désinstaller Trident	`false`
`logFormat`	Format de journalisation Trident à utiliser [TEXT,json]	`"text"`
`tridentImage`	Image Trident à installer	`"netapp/trident:24.10"`

Paramètre	Description	Valeur par défaut
`imageRegistry`	Chemin d'accès au registre interne, du format `<registry FQDN>[:port][/subpath]`	`"k8s.gcr.io"` (Kubernetes 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 une suppression complète de 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.Tolerations`.	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.Tolerations`.	Pas de valeur par défaut ; facultatif
`nodePrep`	Permet à Trident de préparer les nœuds du cluster Kubernetes à gérer les volumes à l'aide du protocole de stockage de données spécifié. **Actuellement, `iscsi` est la seule valeur prise en charge.**	

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

Détails sur le détachement forcé

Forcer le détachement est disponible pour `ontap-san`, `ontap-san-economy` et `onatp-nas-economy` uniquement. Avant d'activer le détachement forcé, l'arrêt non autorisé des nœuds (NGN) doit être activé sur le cluster Kubernetes. Pour plus d'informations, reportez-vous "Kubernetes : arrêt du nœud sans interruption"à .

Lorsque vous utilisez le `ontap-nas-economy` pilote, vous devez définir le `autoExportPolicy` paramètre de la configuration back-end sur `true` afin que Trident puisse restreindre l'accès au nœud Kubernetes avec le taint appliqué à l'aide de règles d'exportation gérées.

Comme Trident repose sur les NGN Kubernetes, ne supprimez pas les `out-of-service` nœuds défectueux avant que toutes les charges de travail non tolérables ne soient replanifiées. L'application ou la suppression imprudemment de cet outil peut compromettre la protection des données back-end.

Lorsque l'administrateur du cluster Kubernetes a appliqué le `node.kubernetes.io/out-of-service=nodeshutdown:NoExecute` taint au nœud et `enableForceDetach` est défini sur `true`, Trident détermine l'état du nœud et :

1. Cessez l'accès aux E/S back-end pour les volumes montés sur ce nœud.
2. Marquer l'objet de nœud Trident comme `dirty` (non sécurisé pour les nouvelles publications).

> Le contrôleur Trident rejette les nouvelles demandes de volume publiées jusqu'à ce que le nœud soit de nouveau qualifié (après avoir été marqué comme `dirty`) par le pod de nœud Trident. Toutes les charges de travail planifiées avec une demande de volume persistant montée (même lorsque le nœud du cluster est sain et prêt) ne seront pas acceptées tant que Trident ne pourra pas vérifier le nœud `clean` (compatibilité pour les nouvelles publications).

Lorsque l'intégrité du nœud est restaurée et que la taint est supprimée, Trident :

1. Identifiez et nettoyez les chemins publiés obsolètes sur le nœud.
2. Si le nœud est dans un `cleanable` état (le taint hors service a été supprimé et le nœud est à `Ready` l'état) et que tous les chemins obsolètes et publiés sont propres, Trident reprépare le nœud en tant que et autorise la publication de `clean` nouveaux volumes sur le nœud.

Exemples de configurations

Vous pouvez utiliser les attributs dans Options de configuration lors de la définition `TridentOrchestrator` pour personnaliser votre installation.

Configuration personnalisée de base

Ceci est un exemple d'installation 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
```

Sélecteurs de nœuds

Cet exemple installe Trident avec des sélecteurs de nœuds.

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

Nœuds worker Windows

Cet exemple installe Trident sur un noeud 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
```

Identités gérées sur un cluster AKS

Cet exemple installe Trident pour activer les identités gérées sur un cluster AKS.

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  cloudProvider: "Azure"
```

Identité cloud sur un cluster AKS

Cet exemple installe Trident pour une utilisation avec une identité de cloud sur un cluster AKS.

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  cloudProvider: "Azure"
  cloudIdentity: 'azure.workload.identity/client-id: xxxxxxxx-xxxx-
xxxx-xxxx-xxxxxxxxxxx'
```

Identité cloud sur un cluster EKS

Cet exemple installe Trident pour une utilisation avec une identité de cloud sur un cluster AKS.

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  cloudProvider: "AWS"
  cloudIdentity: "'eks.amazonaws.com/role-arn:
arn:aws:iam::123456:role/trident-role'"
```

Identité cloud pour GKE

Cet exemple installe Trident pour une utilisation avec une identité de cloud sur un cluster GKE.

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-gcp-gcnv
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: '012345678901'
  network: gcnv-network
  location: us-west2
  serviceLevel: Premium
  storagePool: pool-premium1
```

Installation à l'aide de tridentctl

Installation à l'aide de tridentctl

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

Informations critiques sur Trident 24.10

Vous devez lire les informations critiques suivantes sur Trident.

** informations sur le Trident **

- Kubernetes 1.27 est désormais pris en charge dans Trident. Mise à niveau de Trident avant la mise à niveau de Kubernetes.
- Trident applique strictement l'utilisation de la configuration de chemins d'accès multiples dans les environnements SAN, avec une 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.

Installez 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 d'installation Trident crée un pod Trident, configure les objets CRD utilisés pour maintenir son état et initialise les side-cars 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 sur "La section *Assets* sur GitHub". Mettez à jour *<Trident-installer-XX.XX.X.tar.gz>* dans l'exemple avec la version Trident sélectionnée.

```
wget https://github.com/NetApp/trident/releases/download/v24.10.0/trident-
installer-24.10.0.tar.gz
tar -xf trident-installer-24.10.0.tar.gz
cd trident-installer
```

Étape 2 : installez Trident

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

Mode standard

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

Images dans un registre

```
./tridentctl install -n trident --image-registry <your-registry>
--autosupport-image <your-registry>/trident-autosupport:24.10 --trident
-image <your-registry>/trident:24.10.0
```

Images dans différents registres

```
./tridentctl install -n trident --image-registry <your-registry>
--autosupport-image <your-registry>/trident-autosupport:24.10 --trident
-image <your-registry>/trident:24.10.0
```

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=24.10.0
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 si l'installation de Trident est terminée en vérifiant l'état 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 de Trident installée.

```
./tridentctl -n trident version

+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 24.10.0        | 24.10.0        |
+----------------+----------------+
```

Exemples de configurations

Les exemples suivants fournissent des exemples de configuration pour l'installation de Trident à l'aide de `tridentctl`.

Nœuds Windows

Pour activer l'exécution de Trident sur les nœuds Windows :

```
tridentctl install --windows -n trident
```

Forcer le détachement

Pour plus d'informations sur le détachement forcé, voir "Personnalisez l'installation de l'opérateur Trident".

```
tridentctl install --enable-force-detach=true -n trident
```

Personnalisez l'installation tridentctl

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

En savoir plus sur le programme d'installation

Le programme d'installation de 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 en utilisant `--trident-image`. Si vous avez copié l'image Trident ainsi que les images de side-car CSI nécessaires dans un référentiel privé, il peut être préférable de spécifier l'emplacement de ce référentiel à l'aide du `--image-registry` commutateur, qui prend la forme `<registry FQDN>[:port]`.

> Lors de l'installation de Trident dans un référentiel privé, si vous utilisez le `--image-registry` commutateur pour spécifier l'emplacement du référentiel, n'utilisez pas `/netapp/` dans le chemin du référentiel. Par exemple : `./tridentctl install --image-registry <image-registry> -n <namespace>`

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
```

Utilisez 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 les outils NFS, iSCSI, NVMe/TCP ou FC en fonction de votre sélection de pilotes.

Choisir les bons outils

Si vous utilisez une combinaison de pilotes, vous devez installer tous les outils requis pour vos pilotes. Les versions récentes de RedHat CoreOS ont les outils installés par défaut.

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`.

Outils NVMe

"Installez les outils NVMe" si vous utilisez `ontap-san` Pour le protocole NVMe (Nonvolatile Memory Express) sur TCP (NVMe/TCP).

> ONTAP 9.12 ou version ultérieure est recommandé pour NVMe/TCP.

Outils SCSI sur FC

SCSI over Fibre Channel (FC) est une fonctionnalité de prévisualisation technique dans la version Trident 24.10.

"Installez les outils iSCSI" Si vous utilisez `ontap-san` avec sanType `fcp` (SCSI sur FC).

Pour plus d'informations, reportez-vous à la section "Manières de configurer FC ; hôtes SAN FC-NVMe" .

Détection des services de nœud

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

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

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

Examiner les services découverts

Trident identifie les services activés pour chaque nœud sur le 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

Trident peut automatiquement établir 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, Trident exécute l'autorétablissement 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. Trident détermine les priorités de réparation et le moment où 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 `trident-main` le conteneur sur le pod Demeset respectif. Pour afficher les journaux, vous devez avoir défini `debug` la valeur « true » lors de l'installation de Trident.

Les fonctionnalités d'auto-rétablissement iSCSI de Trident permettent d'éviter :

- Sessions iSCSI obsolètes ou non saines pouvant survenir après un problème de connectivité réseau. Dans le cas d'une session obsolète, 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

Points à prendre en compte avant de mettre à niveau Trident

- Si seuls les igroups par nœud (introduits dans 23.04+) sont utilisés, l'auto-rétablissement iSCSI lance des renumérisations SCSI pour tous les périphériques du bus SCSI.
- Si seuls les igroups dont le périmètre est back-end (obsolète à partir de la version 23.04) sont utilisés, l'auto-rétablissement iSCSI lance des renumérisations SCSI pour obtenir les ID de LUN exacts dans le bus SCSI.
- Si une combinaison d'igroups par nœud et d'igroups scoped back-end est utilisée, l'autorétablissement iSCSI lance des renumérisations SCSI pour obtenir des ID de LUN exacts sur le bus SCSI.

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. Reportez-vous à la section "Documentation Red Hat".

RHEL 8+

1. Installez les packages système suivants :

```
sudo yum install -y lsscsi iscsi-initiator-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. 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`.

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

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

5. 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.

Configurez ou désactivez l'auto-rétablissement iSCSI

Vous pouvez configurer les paramètres d'auto-rétablissement iSCSI Trident suivants pour corriger les sessions obsolètes :

- **Intervalle d'auto-rétablissement iSCSI** : détermine la fréquence à laquelle l'auto-rétablissement iSCSI est appelé (par défaut : 5 minutes). Vous pouvez le configurer pour qu'il s'exécute plus fréquemment en définissant un nombre plus petit ou moins fréquemment en définissant un nombre plus grand.

La définition de l'intervalle d'auto-rétablissement iSCSI sur 0 arrête complètement l'auto-rétablissement iSCSI. Nous ne recommandons pas de désactiver l'auto-rétablissement iSCSI. Il ne doit être désactivé que dans certains cas lorsque l'auto-rétablissement iSCSI ne fonctionne pas comme prévu ou à des fins de débogage.

- **Délai d'attente d'auto-rétablissement iSCSI** : détermine la durée d'attente de l'auto-rétablissement iSCSI avant de se déconnecter d'une session défectueuse et de tenter de se reconnecter (par défaut : 7 minutes). Vous pouvez le configurer sur un nombre plus grand de sorte que les sessions identifiées comme non saines doivent attendre plus longtemps avant d'être déconnectées, puis une tentative de

connexion est faite, ou un nombre plus petit pour se déconnecter et se connecter plus tôt.

Gouvernail

Pour configurer ou modifier les paramètres d'auto-rétablissement iSCSI, passez le `iscsiSelfHealingInterval` et `iscsiSelfHealingWaitTime` paramètres lors de l'installation de helm ou de la mise à jour de helm.

L'exemple suivant définit l'intervalle d'auto-rétablissement iSCSI sur 3 minutes et le temps d'attente d'auto-rétablissement sur 6 minutes :

```
helm install trident trident-operator-100.2410.0.tgz --set
iscsiSelfHealingInterval=3m0s --set iscsiSelfHealingWaitTime=6m0s -n
trident
```

tridentctl

Pour configurer ou modifier les paramètres d'auto-rétablissement iSCSI, passez le `iscsi-self-healing-interval` et `iscsi-self-healing-wait-time` paramètres lors de l'installation ou de la mise à jour de tridentctl.

L'exemple suivant définit l'intervalle d'auto-rétablissement iSCSI sur 3 minutes et le temps d'attente d'auto-rétablissement sur 6 minutes :

```
tridentctl install --iscsi-self-healing-interval=3m0s --iscsi-self
-healing-wait-time=6m0s -n trident
```

Volumes NVMe/TCP

Installez les outils NVMe à l'aide des commandes correspondant à votre système d'exploitation.

- NVMe requiert RHEL 9 ou version ultérieure.
- Si la version du noyau de votre nœud Kubernetes est trop ancienne ou si le package NVMe n'est pas disponible pour votre version du noyau, vous devrez peut-être mettre à jour la version du noyau de votre nœud avec le package NVMe.

RHEL 9

```
sudo yum install nvme-cli
sudo yum install linux-modules-extra-$(uname -r)
sudo modprobe nvme-tcp
```

Ubuntu

```
sudo apt install nvme-cli
sudo apt -y install linux-modules-extra-$(uname -r)
sudo modprobe nvme-tcp
```

Vérifiez l'installation

Après l'installation, vérifiez que chaque nœud du cluster Kubernetes dispose d'un NQN unique via la commande :

```
cat /etc/nvme/hostnqn
```

Trident modifie la `ctrl_device_tmo` valeur pour s'assurer que NVMe ne renonce pas au chemin s'il tombe en panne. Ne modifiez pas ce paramètre.

Prise en charge de Fibre Channel (FC)

Vous pouvez désormais utiliser le protocole Fibre Channel (FC) avec Trident pour provisionner et gérer les ressources de stockage sur un système ONTAP.

SCSI over Fibre Channel (FC) est une fonctionnalité de prévisualisation technique dans la version Trident 24.10.

Fibre Channel est un protocole largement adopté dans les environnements de stockage d'entreprise en raison de ses performances élevées, de sa fiabilité et de son évolutivité. Il fournit un canal de communication robuste et efficace pour les périphériques de stockage, permettant des transferts de données rapides et sécurisés. En utilisant SCSI over Fibre Channel, vous pouvez exploiter leur infrastructure de stockage SCSI existante tout en bénéficiant des performances élevées et des capacités longue distance de Fibre Channel. Il permet de consolider les ressources de stockage et de créer des réseaux de stockage (SAN) évolutifs et efficaces, capables de gérer d'importants volumes de données à faible latence.

La fonctionnalité FC de Trident vous permet d'effectuer les opérations suivantes :

- Provisionnez les demandes de service virtuels de manière dynamique en fonction des spécifications de déploiement.
- Prenez des snapshots de volume et créez un nouveau volume à partir de l'instantané.
- Cloner une FC-PVC existante.
- Redimensionner un volume déjà déployé.

Prérequis

Configurez les paramètres réseau et nœud requis pour FC.

Paramètres réseau

1. Obtenez le WWPN des interfaces cibles. Pour plus d'informations, reportez-vous à la section "interface réseau affiche" .
2. Procurez-vous le WWPN pour les interfaces sur l'initiateur (hôte).

   Reportez-vous aux utilitaires correspondants du système d'exploitation hôte.
3. Configurer la segmentation sur le commutateur FC à l'aide des WWPN de l'hôte et de la cible.

   Pour plus d'informations, reportez-vous à la documentation du fournisseur du commutateur Respecive.

   Pour plus d'informations, reportez-vous à la documentation ONTAP suivante :

   - "Présentation de la segmentation Fibre Channel et FCoE"
   - "Manières de configurer FC ; hôtes SAN FC-NVMe"

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 worker pour FC, vous devez installer les outils requis.

Installez les outils FC

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

- 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. Reportez-vous à la section "Documentation Red Hat".

RHEL 8+

1. Installez les packages système suivants :

```
sudo yum install -y lsscsi iscsi-initiator-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. 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`.

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

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

5. 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.

Créer une configuration back-end

Créez un backend Trident pour le `ontap-san` pilote et `fcp` comme sanType.

Se reporter à :

- "Préparez la configuration du système back-end avec les pilotes SAN ONTAP"
- "Options et exemples de configuration des SAN ONTAP"

Exemple de configuration back-end avec FC

```
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
  sanType: fcp
  svm: trident_svm
  credentials:
    name: backend-tbc-ontap-san-secret
```

Créer une classe de stockage

Pour plus d'informations, se reporter à :

- "Options de configuration du stockage"

Exemple de classe de stockage

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: fcp-sc
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
  protocol: "fcp"
  storagePool: "aggr1"
allowVolumeExpansion: True
```

Configuration et gestion des systèmes back-end

Configuration des systèmes back-end

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

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 Trident avec Amazon FSX pour NetApp ONTAP"

Azure NetApp Files

Configurer un back-end Azure NetApp Files

Vous pouvez configurer Azure NetApp Files en tant que back-end pour Trident. Vous pouvez relier des volumes NFS et SMB à l'aide d'un back-end Azure NetApp Files. Trident prend également en charge la gestion des identifiants à l'aide d'identités gérées pour les clusters Azure Kubernetes Services (AKS).

Détails du pilote Azure NetApp Files

Trident fournit les pilotes de stockage Azure NetApp Files suivants pour communiquer avec le cluster. Les modes d'accès pris en charge sont : *ReadWriteOnce* (RWO), *ReadOnlyMey* (ROX), *ReadWriteMaly* (RWX), *ReadWriteOncePod* (RWOP).

Conducteur	Protocole	Mode Volume	Modes d'accès pris en charge	Systèmes de fichiers pris en charge
`azure-netapp-files`	NFS PME	Système de fichiers	RWO, ROX, RWX, RWOP	`nfs`, `smb`

Considérations

- Le service Azure NetApp Files ne prend pas en charge les volumes inférieurs à 50 Gio. Trident crée automatiquement des volumes de 50 Gio si un volume plus petit est demandé.
- Trident prend en charge les volumes SMB montés sur les pods s'exécutant sur les nœuds Windows uniquement.

Identités gérées pour AKS

Trident prend en charge "identités gérées"les clusters Azure Kubernetes Services. Pour tirer parti de la gestion rationalisée des informations d'identification offerte par les identités gérées, vous devez disposer des éléments suivants :

- Cluster Kubernetes déployé à l'aide d'AKS
- Identités gérées configurées sur le cluster AKS kubernetes
- Trident installé qui inclut le `cloudProvider` à spécifier `"Azure"`.

Opérateur Trident

Pour installer Trident à l'aide de l'opérateur Trident, modifiez `tridentorchestrator_cr.yaml` pour définir sur `cloudProvider "Azure"` . Par exemple :

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  imagePullPolicy: IfNotPresent
  cloudProvider: "Azure"
```

Gouvernail

L'exemple suivant installe les ensembles Trident `cloudProvider` sur Azure à l'aide de la variable d'environnement `$CP` :

```
helm install trident trident-operator-100.2410.0.tgz --create
-namespace --namespace <trident-namespace> --set cloudProvider=$CP
```

<code>tridentctl</code>

L'exemple suivant installe Trident et définit l' `cloudProvider`indicateur sur `Azure`:

```
tridentctl install --cloud-provider="Azure" -n trident
```

Identité cloud pour AKS

L'identité cloud permet aux pods Kubernetes d'accéder aux ressources Azure en s'authentifiant comme identité de workload au lieu de fournir des informations d'identification Azure explicites.

Pour tirer parti de l'identité cloud dans Azure, vous devez disposer des éléments suivants :

- Cluster Kubernetes déployé à l'aide d'AKS
- Identité de la charge de travail et émetteur oidc configurés sur le cluster AKS Kubernetes
- Trident installé, qui inclut le `cloudProvider` à spécifier `"Azure"` et `cloudIdentity` spécifier l'identité de la charge de travail

Opérateur Trident

Pour installer Trident à l'aide de l'opérateur Trident, modifiez `tridentorchestrator_cr.yaml` pour définir sur `cloudProvider "Azure"` et définir `cloudIdentity` sur `azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxx`.

Par exemple :

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  imagePullPolicy: IfNotPresent
  cloudProvider: "Azure"
  *cloudIdentity: 'azure.workload.identity/client-id: xxxxxxxx-xxxx-
xxxx-xxxx-xxxxxxxxxxx'*
```

Gouvernail

Définissez les valeurs des indicateurs **cloud-Provider (CP)** et **cloud-Identity (ci)** à l'aide des variables d'environnement suivantes :

```
export CP="Azure"
export CI="'azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxxx'"
```

L'exemple suivant installe Trident et définit `cloudProvider` dans Azure à l'aide de la variable d'environnement `$CP` et définit `cloudIdentity` à l'aide de la variable d'environnement `$CI` :

```
helm install trident trident-operator-100.2410.0.tgz --set
cloudProvider=$CP --set cloudIdentity="$CI"
```

<code>tridentctl</code>

Définissez les valeurs des indicateurs **cloud Provider** et **cloud Identity** à l'aide des variables d'environnement suivantes :

```
export CP="Azure"
export CI="azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxxx"
```

L'exemple suivant installe Trident et définit l' `cloud-provider`indicateur sur `$CP`, et `cloud-identity` sur `$CI`:

```
tridentctl install --cloud-provider=$CP --cloud-identity="$CI" -n
trident
```

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.

Prérequis pour les volumes NFS et SMB

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".

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

> - `subscriptionID`, `tenantID`, `clientID`, `location`, et `clientSecret` Sont facultatives lors de l'utilisation d'identités gérées sur un cluster AKS.
> - `tenantID`, `clientID`, et `clientSecret` Sont facultatives lors de l'utilisation d'une identité de cloud sur un cluster AKS.

- 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 (`assignableScopes`l'abonnement) avec les autorisations suivantes qui sont limitées à ce que Trident exige. Après avoir créé le rôle personnalisé, "Attribuez le rôle à l'aide du portail Azure".

Rôle de contributeur personnalisé

```
{
    "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/
write",

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

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/MountTarge
ts/read",
                    "Microsoft.Network/virtualNetworks/read",

"Microsoft.Network/virtualNetworks/subnets/read",

"Microsoft.Features/featureProviders/subscriptionFeatureRegistrat
ions/read",

"Microsoft.Features/featureProviders/subscriptionFeatureRegistrat
```

```
ions/write",

"Microsoft.Features/featureProviders/subscriptionFeatureRegistrat
ions/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.
- À utiliser `Cloud Identity`, obtenir le `client ID` a partir d'un "identité gérée attribuée par l'utilisateur" Et spécifiez cet ID dans `azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxx`.

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 2022. Trident prend en charge les volumes SMB montés sur les pods s'exécutant sur les nœuds Windows uniquement.
- Au moins un secret Trident contenant vos informations d'identification Active Directory afin 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 NFS et SMB backend pour Azure NetApp Files et passez en revue les exemples de configuration.

Options de configuration du back-end

Trident utilise votre configuration back-end (sous-réseau, réseau virtuel, niveau de service et emplacement) pour créer des volumes Azure NetApp Files sur des pools de capacité disponibles à l'emplacement souhaité et qui correspondent au niveau de service et au sous-réseau requis.

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

Les systèmes Azure NetApp Files back-end 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 Facultatif lorsque les identités gérées sont activées sur un cluster AKS.	
`tenantID`	ID locataire d'un enregistrement d'application Facultatif lorsque des identités gérées ou des identités de cloud sont utilisées sur un cluster AKS.	
`clientID`	L'ID client d'un enregistrement d'application Facultatif lorsque des identités gérées ou des identités de cloud sont utilisées sur un cluster AKS.	
`clientSecret`	Secret client d'un enregistrement d'application Facultatif lorsque des identités gérées ou des identités de cloud sont utilisées sur un cluster AKS.	
`serviceLevel`	Un de `Standard`, `Premium`, ou `Ultra`	« » (aléatoire)

Paramètre	Description	Valeur par défaut
`location`	Nom de l'emplacement Azure dans lequel les nouveaux volumes seront créés Facultatif lorsque les identités gérées sont activées sur un cluster AKS.	
`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)

Paramètre	Description	Valeur 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
`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`
`supportedTopologies`	Représente une liste de régions et de zones prises en charge par ce back-end. Pour plus d'informations, reportez-vous "Utiliser la topologie CSI"à .	

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 trouvé » lors de la création d'une demande de volume persistant, il est probable que votre enregistrement d'application ne dispose pas des autorisations et des ressources requises (sous-réseau, réseau virtuel, pool de capacité). Si le débogage est activé, 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	« True » pour NFSv4 « false » pour NFSv3
`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

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.

Configuration minimale

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

Cette configuration est idéale lorsque vous commencez à utiliser Azure NetApp Files et que vous essayez d'autres fonctionnalités, mais dans la pratique, vous voudrez ajouter de l'étendue aux volumes que vous provisionnez.

```
---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-anf-1
  namespace: trident
spec:
  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
```

Identités gérées pour AKS

Cette configuration back-end omet `subscriptionID`, `tenantID`, `clientID`, et `clientSecret`, qui sont facultatives lors de l'utilisation d'identités gérées.

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-anf-1
  namespace: trident
spec:
  version: 1
  storageDriverName: azure-netapp-files
  capacityPools: ["ultra-pool"]
  resourceGroups: ["aks-ami-eastus-rg"]
  netappAccounts: ["smb-na"]
  virtualNetwork: eastus-prod-vnet
  subnet: eastus-anf-subnet
```

Identité cloud pour AKS

Cette configuration back-end omet `tenantID`, `clientID`, et `clientSecret`, qui sont facultatives lors de l'utilisation d'une identité de nuage.

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-anf-1
  namespace: trident
spec:
  version: 1
  storageDriverName: azure-netapp-files
  capacityPools: ["ultra-pool"]
  resourceGroups: ["aks-ami-eastus-rg"]
  netappAccounts: ["smb-na"]
  virtualNetwork: eastus-prod-vnet
  subnet: eastus-anf-subnet
  location: eastus
  subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
```

Configuration de niveau de service spécifique avec filtres de pool de capacité

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

```
---
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
```

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'
```

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
```

Configuration des topologies prises en charge

Trident facilite le provisionnement des volumes pour les workloads en fonction des régions et des zones de disponibilité. Le `supportedTopologies` bloc de cette configuration back-end est utilisé pour fournir une liste de régions et de zones par back-end. Les valeurs de région et de zone spécifiées ici doivent correspondre aux valeurs de région et de zone indiquées sur les étiquettes de chaque nœud de cluster Kubernetes. Ces régions et zones représentent la liste des valeurs autorisées pouvant être fournies dans une classe de stockage. Pour les classes de stockage qui contiennent un sous-ensemble des régions et zones fournies dans un back-end, Trident crée des volumes dans la région et la zone mentionnées. Pour plus d'informations, reportez-vous "Utiliser la topologie CSI"à .

```
---
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
supportedTopologies:
- topology.kubernetes.io/region: eastus
  topology.kubernetes.io/zone: eastus-1
- topology.kubernetes.io/region: eastus
  topology.kubernetes.io/zone: eastus-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.

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"
```

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}
```

Utilisation de secrets différents 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.

Google Cloud NetApp volumes

Configurez un système back-end Google Cloud NetApp volumes

Vous pouvez désormais configurer Google Cloud NetApp volumes en tant que back-end pour Trident. Vous pouvez relier des volumes NFS à l'aide d'un système back-end Google Cloud NetApp volumes.

Détails du pilote Google Cloud NetApp volumes

Trident fournit le `google-cloud-netapp-volumes` pilote pour communiquer avec le cluster. Les modes d'accès pris en charge sont : *ReadWriteOnce* (RWO), *ReadOnlyMey* (ROX), *ReadWriteMaly* (RWX), *ReadWriteOncePod* (RWOP).

Conducteur	Protocole	Mode Volume	Modes d'accès pris en charge	Systèmes de fichiers pris en charge
`google-cloud-netapp-volumes`	NFS	Système de fichiers	RWO, ROX, RWX, RWOP	`nfs`

Identité cloud pour GKE

L'identité cloud permet aux pods Kubernetes d'accéder aux ressources Google Cloud en s'authentifiant comme identité de workload au lieu de fournir des informations d'identification Google Cloud explicites.

Pour tirer parti de l'identité cloud dans Google Cloud, vous devez disposer des éléments suivants :

- Cluster Kubernetes déployé à l'aide de GKE.
- Identité de la charge de travail et émetteur oidc configurés sur le cluster GKE.
- Trident installé, qui inclut le `cloudProvider` à spécifier `"GCP"` et `cloudIdentity` spécifier l'identité de

la charge de travail.

Opérateur Trident

Pour installer Trident à l'aide de l'opérateur Trident, modifiez `tridentorchestrator_cr.yaml` pour définir sur `cloudProvider "GCP"` et définir `cloudIdentity` sur `iam.gke.io/gcp-service-account: cloudvolumes-admin-sa@mygcpproject.iam.gserviceaccount.com`.

Par exemple :

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  imagePullPolicy: IfNotPresent
  cloudProvider: "GCP"
  cloudIdentity: 'iam.gke.io/gcp-service-account: cloudvolumes-
admin-sa@mygcpproject.iam.gserviceaccount.com'
```

Gouvernail

Définissez les valeurs des indicateurs **cloud-Provider (CP)** et **cloud-Identity (ci)** à l'aide des variables d'environnement suivantes :

```
export CP="GCP"
export ANNOTATION="iam.gke.io/gcp-service-account: cloudvolumes-admin-
sa@mygcpproject.iam.gserviceaccount.com"
```

L'exemple suivant installe Trident et définit `cloudProvider` GCP à l'aide de la variable d'environnement `$CP` et définit le `cloudIdentity` à l'aide de la variable d'environnement `$ANNOTATION` :

```
helm install trident trident-operator-100.2406.0.tgz --set
cloudProvider=$CP --set cloudIdentity="$ANNOTATION"
```

<code>tridentctl</code>

Définissez les valeurs des indicateurs **cloud Provider** et **cloud Identity** à l'aide des variables d'environnement suivantes :

```
export CP="GCP"
export ANNOTATION="iam.gke.io/gcp-service-account: cloudvolumes-admin-
sa@mygcpproject.iam.gserviceaccount.com"
```

L'exemple suivant installe Trident et définit l' `cloud-provider`indicateur sur `$CP`, et `cloud-identity` sur `$ANNOTATION`:

```
tridentctl install --cloud-provider=$CP --cloud
-identity="$ANNOTATION" -n trident
```

Préparez la configuration d'un système back-end Google Cloud NetApp volumes

Avant de pouvoir configurer votre système back-end Google Cloud NetApp volumes, vous devez vous assurer que les exigences suivantes sont respectées.

Prérequis pour les volumes NFS

Si vous utilisez Google Cloud NetApp volumes pour la première fois ou dans un nouvel emplacement, une configuration initiale est requise pour configurer Google Cloud NetApp volumes et créer un volume NFS. Reportez-vous à la "Avant de commencer".

Vérifiez les points suivants avant de configurer le système back-end Google Cloud NetApp volumes :

- Compte Google Cloud configuré avec le service Google Cloud NetApp volumes. Reportez-vous à la "Google Cloud NetApp volumes".
- Numéro de projet de votre compte Google Cloud. Reportez-vous à la "Identification des projets".
- Un compte de service Google Cloud avec le rôle d'administrateur NetApp volumes (`netappcloudvolumes.admin`). Reportez-vous à la "Rôles et autorisations de gestion des identités et des accès".
- Fichier de clé API pour votre compte GCNV. Reportez-vous à "Créez une clé de compte de service"
- Un pool de stockage. Reportez-vous à la "Présentation des pools de stockage".

Pour plus d'informations sur la configuration de l'accès à Google Cloud NetApp volumes, consultez "Configurez l'accès à Google Cloud NetApp volumes"la .

Options et exemples de configuration back-end de Google Cloud NetApp volumes

Découvrez les options de configuration NFS backend pour Google Cloud NetApp volumes et examinez des exemples de configuration.

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	La valeur de `storageDriverName` doit être indiquée comme « google-cloud-netapp-volumes ».

Paramètre	Description	Valeur par défaut
`backendName`	(Facultatif) Nom personnalisé du système back-end de stockage	Nom du pilote + "_" + partie de la clé API
`storagePools`	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.	
`location`	Emplacement Google Cloud dans lequel Trident crée des volumes GCNV. Lors de la création de clusters Kubernetes répartis entre régions, les volumes créés dans un `location` peuvent être utilisés dans des workloads planifiés sur des nœuds répartis sur plusieurs régions Google Cloud. Le trafic entre les régions coûte plus cher.	
`apiKey`	Clé d'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). Le `apiKey` doit inclure des paires clé-valeur pour les clés suivantes : `type` `project_id`, `client_email`, , `client_id`, , , `auth_uri` `token_uri` `auth_provider_x509_cert_url`, et `client_x509_cert_url`.	
`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 d'un pool de stockage et de ses volumes. Les valeurs sont `flex`, `standard`, , `premium`ou `extreme`.	
`network`	Réseau Google Cloud utilisé pour les volumes GCNV.	
`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
`supportedTopologies`	Représente une liste de régions et de zones prises en charge par ce back-end. Pour plus d'informations, reportez-vous "Utiliser la topologie CSI"à . Par exemple : `supportedTopologies: - topology.kubernetes.io/region: asia-east1 topology.kubernetes.io/zone: asia-east1-a`	

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 de toute combinaison d'adresses IPv4.	« 0.0.0.0/0 »
`snapshotDir`	Accès au `.snapshot` répertoire	« True » pour NFSv4 « false » pour NFSv3
`snapshotReserve`	Pourcentage de volume réservé pour les snapshots	« » (accepter la valeur par défaut de 0)
`unixPermissions`	Les autorisations unix des nouveaux volumes (4 chiffres octaux).	« »

Exemples de configurations

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.

Configuration minimale

Il s’agit de la configuration back-end minimale absolue. Avec cette configuration, Trident détecte tous vos pools de stockage délégués aux volumes Google Cloud NetApp dans l’emplacement configuré et place de nouveaux volumes dans l’un de ces pools de manière aléatoire. Comme `nasType` est omis, la `nfs` valeur par défaut s’applique et le back-end provisionne les volumes NFS.

Cette configuration est idéale lorsque vous n’utilisez que Google Cloud NetApp volumes et que vous essayez d’effectuer des opérations. Dans la pratique, vous devrez probablement fournir une étendue supplémentaire pour les volumes que vous provisionnez.

```
---

apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-gcnv-secret
type: Opaque
stringData:
  private_key_id: 'f2cb6ed6d7cc10c453f7d3406fc700c5df0ab9ec'
  private_key: |
    -----BEGIN PRIVATE KEY-----\n
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\n
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\n
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\n
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\n
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\n
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\n
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\n
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\n
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\n
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\n
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\n
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\n
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\n
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\n
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\n
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\n
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\n
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\n
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\n
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\n
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\n
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\n
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\n
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\n
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m\n
```

```
    XsYg6gyxy4zq7OlwWgLwGa==\n
    -----END PRIVATE KEY-----\n

---

apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-gcnv
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: '123455380079'
  location: europe-west6
  serviceLevel: premium
  apiKey:
    type: service_account
    project_id: my-gcnv-project
    client_email: myproject-prod@my-gcnv-
project.iam.gserviceaccount.com
    client_id: '103346282737811234567'
    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/myproject-prod%40my-
gcnv-project.iam.gserviceaccount.com
  credentials:
    name: backend-tbc-gcnv-secret
```

Configuration avec filtre StoragePools

```
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-gcnv-secret
type: Opaque
stringData:
  private_key_id: 'f2cb6ed6d7cc10c453f7d3406fc700c5df0ab9ec'
  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
    XsYg6gyxy4zq7OlwWgLwGa==
    -----END PRIVATE KEY-----

---

apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-gcnv
spec:
```

```
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: '123455380079'
  location: europe-west6
  serviceLevel: premium
  storagePools:
  - premium-pool1-europe-west6
  - premium-pool2-europe-west6
  apiKey:
    type: service_account
    project_id: my-gcnv-project
    client_email: myproject-prod@my-gcnv-
project.iam.gserviceaccount.com
    client_id: '103346282737811234567'
    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/myproject-prod%40my-
gcnv-project.iam.gserviceaccount.com
  credentials:
    name: backend-tbc-gcnv-secret
```

Configuration de pool virtuel

Cette configuration back-end définit plusieurs pools virtuels dans un seul fichier. Les pools virtuels sont définis dans la `storage` section. Ces fonctionnalités sont utiles lorsque plusieurs pools de stockage prennent en charge différents niveaux de services et que vous souhaitez créer dans Kubernetes des classes de stockage qui les représentent. Les étiquettes de pool virtuel sont utilisées pour différencier les pools. Par exemple, dans l'exemple ci-dessous `performance` , le libellé et `serviceLevel` le type sont utilisés pour différencier les pools virtuels.

Vous pouvez également définir des valeurs par défaut applicables à tous les pools virtuels et remplacer les valeurs par défaut des pools virtuels individuels. Dans l'exemple suivant, `snapshotReserve` et `exportRule` servent de valeurs par défaut pour tous les pools virtuels.

Pour plus d'informations, reportez-vous "Pools virtuels"à .

```
---

apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-gcnv-secret
type: Opaque
stringData:
  private_key_id: 'f2cb6ed6d7cc10c453f7d3406fc700c5df0ab9ec'
  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
    XsYg6gyxy4zq7OlwWgLwGa==
    -----END PRIVATE KEY-----

---

apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-gcnv
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: '123455380079'
  location: europe-west6
  apiKey:
    type: service_account
    project_id: my-gcnv-project
    client_email: myproject-prod@my-gcnv-
project.iam.gserviceaccount.com
    client_id: '103346282737811234567'
    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/myproject-prod%40my-
gcnv-project.iam.gserviceaccount.com
  credentials:
    name: backend-tbc-gcnv-secret
  defaults:
    snapshotReserve: '10'
    exportRule: 10.0.0.0/24
  storage:
    - labels:
        performance: extreme
      serviceLevel: extreme
      defaults:
        snapshotReserve: '5'
        exportRule: 0.0.0.0/0
    - labels:
        performance: premium
      serviceLevel: premium
    - labels:
```

```
        performance: standard
      serviceLevel: standard
```

Identité cloud pour GKE

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-gcp-gcnv
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: '012345678901'
  network: gcnv-network
  location: us-west2
  serviceLevel: Premium
  storagePool: pool-premium1
```

Configuration des topologies prises en charge

Trident facilite le provisionnement des volumes pour les workloads en fonction des régions et des zones de disponibilité. Le `supportedTopologies` bloc de cette configuration back-end est utilisé pour fournir une liste de régions et de zones par back-end. Les valeurs de région et de zone spécifiées ici doivent correspondre aux valeurs de région et de zone indiquées sur les étiquettes de chaque nœud de cluster Kubernetes. Ces régions et zones représentent la liste des valeurs autorisées pouvant être fournies dans une classe de stockage. Pour les classes de stockage qui contiennent un sous-ensemble des régions et zones fournies dans un back-end, Trident crée des volumes dans la région et la zone mentionnées. Pour plus d'informations, reportez-vous "Utiliser la topologie CSI"à .

```
---
version: 1
storageDriverName: google-cloud-netapp-volumes
subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
tenantID: 68e4f836-edc1-fake-bff9-b2d865ee56cf
clientID: dd043f63-bf8e-fake-8076-8de91e5713aa
clientSecret: SECRET
location: asia-east1
serviceLevel: flex
supportedTopologies:
- topology.kubernetes.io/region: asia-east1
  topology.kubernetes.io/zone: asia-east1-a
- topology.kubernetes.io/region: asia-east1
  topology.kubernetes.io/zone: asia-east1-b
```

Et la suite ?

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

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

Pour vérifier que le back-end est correctement créé, exécutez la commande suivante :

```
kubectl get tridentbackendconfig

NAME                BACKEND NAME        BACKEND UUID
PHASE   STATUS
backend-tbc-gcnv    backend-tbc-gcnv    b2fd1ff9-b234-477e-88fd-713913294f65
Bound   Success
```

Si la création du back-end échoue, la configuration du back-end est erronée. Vous pouvez décrire le back-end à l'aide de la `kubectl get tridentbackendconfig <backend-name>` commande ou 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 supprimer le back-end et exécuter à nouveau la commande create.

Plus d'exemples

Exemples de définition de classe de stockage

Voici une définition de base `StorageClass` qui fait référence au back-end ci-dessus.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gcnv-nfs-sc
provisioner: csi.trident.netapp.io
parameters:
  backendType: "google-cloud-netapp-volumes"
```

Exemples de définitions utilisant le `parameter.selector` champ :

A l'aide de `parameter.selector` , vous pouvez spécifier pour chaque `StorageClass` système "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: extreme-sc
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=extreme"
  backendType: "google-cloud-netapp-volumes"
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: premium-sc
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=premium"
  backendType: "google-cloud-netapp-volumes"
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: standard-sc
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=standard"
  backendType: "google-cloud-netapp-volumes"
```

Pour plus de détails sur les classes de stockage, reportez-vous "Créer une classe de stockage"à la section .

Exemple de définition de PVC

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: gcnv-nfs-pvc
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 100Gi
  storageClassName: gcnv-nfs-sc
```

Pour vérifier si la demande de volume persistant est liée, exécutez la commande suivante :

```
kubectl get pvc gcnv-nfs-pvc

NAME          STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS AGE
gcnv-nfs-pvc  Bound    pvc-b00f2414-e229-40e6-9b16-ee03eb79a213  100Gi
RWX            gcnv-nfs-sc  1m
```

Configurer un système Cloud Volumes Service pour Google Cloud backend

Découvrez comment configurer NetApp Cloud Volumes Service pour Google Cloud en tant que back-end pour votre installation Trident à l'aide des exemples de configurations fournis.

Détails du pilote Google Cloud

Trident fournit le `gcp-cvs` pilote pour communiquer avec le cluster. Les modes d'accès pris en charge sont : *ReadWriteOnce* (RWO), *ReadOnlyMey* (ROX), *ReadWriteMaly* (RWX), *ReadWriteOncePod* (RWOP).

Conducteur	Protocole	Mode Volume	Modes d'accès pris en charge	Systèmes de fichiers pris en charge
`gcp-cvs`	NFS	Système de fichiers	RWO, ROX, RWX, RWOP	`nfs`

En savoir plus sur la prise en charge de Trident pour Cloud Volumes Service pour Google Cloud

Trident peut créer des volumes Cloud Volumes Service en deux "types de service":

- **CVS-Performance** : type de service 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 "trois niveaux de service"options suivantes :
  - `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. Utilisez `software` pour sélectionner le type de service CVS. Sinon, Trident suppose le 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 Trident crée des volumes Cloud Volumes Service. Lors de la création de clusters Kubernetes répartis entre régions, les volumes créés dans un `apiRegion` peuvent être utilisés dans des workloads planifiés sur des nœuds répartis sur plusieurs régions Google Cloud. Le trafic entre les régions coûte plus cher.	
`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).	

Paramètre	Description	Valeur par défaut
`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

`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)

Paramètre	Description	Valeur par défaut
`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-----
    <key_value>
    -----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-----
    <key_value>
    -----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-----
    <key_value>
    -----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: csi.trident.netapp.io
parameters:
  selector: "performance=extreme; protection=extra"
allowVolumeExpansion: true
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: cvs-extreme-standard-protection
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=premium; protection=standard"
allowVolumeExpansion: true
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: cvs-premium-extra-protection
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=premium; protection=extra"
allowVolumeExpansion: true
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: cvs-premium
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=premium; protection=standard"
allowVolumeExpansion: true
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: cvs-standard
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=standard"
allowVolumeExpansion: true
```

```
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: cvs-extra-protection
provisioner: csi.trident.netapp.io
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`) fait appel à n'importe quel pool de stockage qui fournit une réserve de snapshots de 10 %. Trident détermine quel pool virtuel est sélectionné et veille à ce que les exigences de réserve d'instantanés soient 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-----
    <key_value>
    -----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-----
    <key_value>
    -----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 back-end Element avec votre installation Trident.

Détails du pilote d'élément

Trident fournit le `solidfire-san` pilote de stockage pour communiquer avec le cluster. Les modes d'accès pris en charge sont : *ReadWriteOnce* (RWO), *ReadOnlyMey* (ROX), *ReadWriteMaly* (RWX), *ReadWriteOncePod* (RWOP).

Le `solidfire-san` pilote de stockage prend en charge les modes *file* et *block* volume. Pour le `Filesystem` volumeMode, Trident crée un volume et 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, RWOP	Aucun système de fichiers. Périphérique de bloc brut.
`solidfire-san`	ISCSI	Système de fichiers	RWO, RWOP	`xfs`, `ext3`, `ext4`

Avant de commencer

Vous aurez besoin des éléments suivants avant de créer un back-end d'élément.

- 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. Reportez-vous à la section "informations de préparation du nœud de travail".

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 ».

Paramètre	Description	Valeur par défaut
`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)	
`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. Trident utilise CHAP.	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.

Trident copie les étiquettes présentes sur un pool de stockage vers la LUN de stockage back-end au moment 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 d'entre eux 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`) est mappée sur le premier pool virtuel. Il s'agit de la seule piscine offrant des performances or avec un `Volume Type QoS` de Gold. La dernière classe de stockage (`solidfire-silver`) fait référence à n'importe quel pool de stockage offrant des performances Silver. Trident décide du pool virtuel sélectionné et s'assure 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"

Pilotes SAN de ONTAP

Présentation du pilote SAN ONTAP

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

Détails du pilote SAN ONTAP

Trident fournit les pilotes de stockage SAN suivants pour communiquer avec le cluster ONTAP. Les modes d'accès pris en charge sont : *ReadWriteOnce* (RWO), *ReadOnlyMey* (ROX), *ReadWriteMaly* (RWX), *ReadWriteOncePod* (RWOP).

Conducteur	Protocole	Mode Volume	Modes d'accès pris en charge	Systèmes de fichiers pris en charge
`ontap-san`	ISCSI	Bloc	RWO, ROX, RWX, RWOP	Pas de système de fichiers, périphérique de bloc brut
`ontap-san`	ISCSI	Système de fichiers	RWO, RWOP ROX et RWX ne sont pas disponibles en mode de volume du système de fichiers.	`xfs`, `ext3`, `ext4`
`ontap-san`	NVMe/TCP Reportez-vous à la section Autres considérations relatives au NVMe/TCP.	Bloc	RWO, ROX, RWX, RWOP	Pas de système de fichiers, périphérique de bloc brut
`ontap-san`	NVMe/TCP Reportez-vous à la section Autres considérations relatives au NVMe/TCP.	Système de fichiers	RWO, RWOP ROX et RWX ne sont pas disponibles en mode de volume du système de fichiers.	`xfs`, `ext3`, `ext4`

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

- Utiliser `ontap-san-economy` uniquement si le nombre d'utilisations du volume persistant doit être supérieur à "Limites de volume ONTAP prises en charge".
- Utiliser `ontap-nas-economy` uniquement si le nombre d'utilisations du volume persistant doit être supérieur à "Limites de volume ONTAP prises en charge" 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

Trident s'attend à être exécuté en tant qu'administrateur ONTAP ou SVM, en général avec l'utilisateur du cluster ou un `vsadmin` utilisateur SVM, ou en tant qu' `admin`utilisateur avec un nom différent et le même rôle. Pour les déploiements Amazon FSX pour NetApp ONTAP, Trident prévoit d'être exécuté en tant qu'administrateur ONTAP ou SVM, en utilisant l'utilisateur du cluster `fsxadmin` ou un `vsadmin` utilisateur SVM, ou un utilisateur avec un nom différent ayant le même rôle. `fsxadmin`L'utilisateur est un remplaçant limité pour l'utilisateur admin du cluster.

Si vous utilisez le `limitAggregateUsage` paramètre, les autorisations d'administration du cluster sont requises. Lors de l'utilisation d'Amazon FSX for NetApp ONTAP avec Trident, le `limitAggregateUsage` paramètre ne fonctionnera pas avec les `vsadmin` comptes d'utilisateur et `fsxadmin`. L'opération de configuration échoue si vous spécifiez ce paramètre.

S'il est possible de créer au sein de ONTAP un rôle plus restrictif 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.

Autres considérations relatives au NVMe/TCP

Trident prend en charge le protocole NVMe (non-volatile Memory Express) avec le `ontap-san` pilote, notamment :

- IPv6
- Copies Snapshot et clones de volumes NVMe
- Redimensionnement d'un volume NVMe
- Importation d'un volume NVMe créé en dehors de Trident afin que son cycle de vie puisse être géré par

Trident

- Chemins d'accès multiples natifs NVMe
- Arrêt normal ou sans gracieuse des nœuds K8s (24.06)

Trident ne prend pas en charge :

- DH-HMAC-CHAP pris en charge par NVMe de manière native
- Chemins d'accès multiples du mappeur de périphériques (DM)
- Cryptage LUKS

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

Découvrez les exigences et les options d'authentification pour la configuration d'un back-end ONTAP avec des pilotes SAN ONTAP.

De formation

Pour tous les systèmes ONTAP back-end, Trident requiert au moins un agrégat attribué 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. Reportez-vous à la section "Préparez le nœud de travail" pour plus d'informations.

Authentifiez le back-end ONTAP

Trident propose deux modes d'authentification d'un 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 un certificat : Trident peut également communiquer avec un cluster ONTAP à l'aide d'un certificat installé sur le 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

Trident exige que les identifiants d'un administrateur SVM-scoped/cluster-scoped communiquent avec le back-

end ONTAP. Il est recommandé d'utiliser des rôles standard prédéfinis tels que `admin` ou `vsadmin`. La compatibilité avec les futures versions d'ONTAP qui exposent les API de fonctionnalités à utiliser dans les futures versions d'Trident est ainsi garantie. Un rôle de connexion de sécurité personnalisé peut être créé et utilisé avec Trident, mais il n'est pas recommandé.

Voici un exemple de définition du back-end :

YAML

```
---
version: 1
backendName: ExampleBackend
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.

Activer l'authentification basée sur certificat

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 back-end réussie indique que Trident peut communiquer avec le back-end ONTAP et gérer les futures opérations de volume.

Créez un rôle ONTAP personnalisé pour Trident

Vous pouvez créer un rôle de cluster ONTAP avec une Privileges minimale afin de ne pas avoir à utiliser le rôle ONTAP admin pour effectuer des opérations dans Trident. Lorsque vous incluez le nom d'utilisateur dans une configuration Trident backend, Trident utilise le rôle de cluster ONTAP que vous avez créé pour effectuer les opérations.

Pour plus d'informations sur la création de rôles personnalisés Trident, reportez-vous à la section"Générateur de rôle personnalisé Trident".

Utilisation de l'interface de ligne de commandes ONTAP

1. Créez un rôle à l'aide de la commande suivante :

```
security login role create <role_name\> -cmddirname "command" -access all
–vserver <svm_name\>
```

2. Créez un nom d'utilisateur pour l'utilisateur Trident :

```
security login create -username <user_name\> -application ontapi
-authmethod <password\> -role <name_of_role_in_step_1\> –vserver
<svm_name\> -comment "user_description"
```

3. Mapper le rôle à l'utilisateur :

```
security login modify username <user_name\> –vserver <svm_name\> -role
<role_name\> -application ontapi -application console -authmethod
<password\>
```

À l'aide de System Manager

Dans ONTAP System Manager, effectuez les opérations suivantes :

1. **Créer un rôle personnalisé** :
   a. Pour créer un rôle personnalisé au niveau du cluster, sélectionnez **Cluster > Paramètres**.

      (Ou) pour créer un rôle personnalisé au niveau du SVM, sélectionner **stockage > Storage VM > >> Paramètres > `required SVM` utilisateurs et rôles**.

   b. Sélectionnez l'icône de flèche (→) en regard de **utilisateurs et rôles**.
   c. Sélectionnez **+Ajouter** sous **rôles**.
   d. Définissez les règles du rôle et cliquez sur **Enregistrer**.
2. **Mapper le rôle à l'utilisateur Trident**: + effectuez les étapes suivantes sur la page **utilisateurs et rôles** :
   a. Sélectionnez Ajouter l'icône **+** sous **utilisateurs**.
   b. Sélectionnez le nom d'utilisateur requis et sélectionnez un rôle dans le menu déroulant pour **role**.
   c. Cliquez sur **Enregistrer**.

Pour plus d'informations, reportez-vous aux pages suivantes :

- "Rôles personnalisés pour l'administration de ONTAP" ou "Définissez des rôles personnalisés"
- "Travaillez avec les rôles et les utilisateurs"

Authentifier les connexions avec le protocole CHAP bidirectionnel

Trident peut authentifier les sessions iSCSI avec le protocole CHAP bidirectionnel pour les `ontap-san` pilotes et `ontap-san-economy`. Pour ce faire, vous devez activer `useCHAP` l'option dans votre définition de back-end. Lorsque ce paramètre est défini sur `true`, Trident configure la sécurité initiateur par défaut du SVM sur CHAP bidirectionnel et définit le nom d'utilisateur et les secrets à partir du fichier back-end. 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
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

En définissant la `useCHAP` valeur sur true, l'administrateur du stockage demande à Trident de configurer CHAP sur le back-end de stockage. Ceci inclut les éléments suivants :

- Configuration du protocole CHAP sur le SVM :
  - Si le type de sécurité initiateur par défaut du SVM est none (défini par défaut) **et** il n'y a pas de LUN préexistantes déjà présentes dans le volume, Trident définit le type de sécurité par défaut sur `CHAP` et passe à la configuration de l'initiateur CHAP et du nom d'utilisateur et des secrets cible.
  - Si le SVM contient des LUN, Trident n'activera pas CHAP sur le SVM. Cela garantit que l'accès aux LUNs 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).

Une fois le back-end créé, Trident crée un code CRD correspondant `tridentbackend` et stocke les secrets CHAP et les noms d'utilisateur comme secrets Kubernetes. Tous les volumes persistants créés par Trident sur ce back-end seront montés et rattachés via 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 backend, 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 commandes/ONTAP, car Trident ne pourra pas récupérer 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",
    "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 continueront à rester actives si les informations d'identification sont mises à jour par Trident sur le SVM. Les nouvelles connexions utilisent les informations d'identification mises à jour et les connexions existantes restent 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 les pilotes SAN ONTAP avec votre installation Trident. Cette section fournit des exemples de configuration 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-economy`, `ontap-nas-flexgroup`, `ontap-san`, `ontap-san-economy`
`backendName`	Nom personnalisé ou système back-end de stockage	Nom du pilote + "_" + dataLIF
`managementLIF`	Adresse IP d'un cluster ou d'une LIF de management du SVM. Un nom de domaine complet (FQDN) peut être spécifié. Peut être configuré pour utiliser des adresses IPv6 si Trident a été installé à l'aide de l'indicateur IPv6. Les adresses IPv6 doivent être définies entre crochets, par exemple `[28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]`. Pour un basculement MetroCluster transparent, consultez le [mcc-best].	« 10.0.0.1 », « [2001:1234:abcd::fefe] »
`dataLIF`	Adresse IP de la LIF de protocole. **Ne spécifiez pas pour iSCSI.** Trident utilise "Mappage de LUN sélectif ONTAP" pour découvrir les LIF iSCI nécessaires à l'établissement d'une session à chemins multiples. Un avertissement est généré si `dataLIF` est explicitement défini. **Omettre pour MetroCluster.** Voir la [mcc-best].	Dérivé par la SVM
`svm`	Serveur virtuel de stockage à utiliser **Omettre pour MetroCluster.** Voir [mcc-best].	Dérivé d'un SVM `managementLIF` est spécifié
`useCHAP`	Utilisez CHAP pour authentifier iSCSI pour les pilotes SAN ONTAP [Boolean]. Set to `true` for Trident to configure and use bidirectionnelle CHAP as the default Authentication for the SVM donné au back-end. Voir "Préparez la configuration du système back-end avec les pilotes SAN ONTAP" pour plus de détails.	`false`
`chapInitiatorSecret`	Secret de l'initiateur CHAP. Requis si `useCHAP=true`	« »
`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`	« »

Paramètre	Description	Valeur par défaut
`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
`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`
`aggregate`	Agrégat pour le provisionnement (facultatif ; si défini, doit être attribué au SVM) Pour le `ontap-nas-flexgroup` pilote, cette option est ignorée. S'ils ne sont pas affectés, les agrégats disponibles peuvent être utilisés pour provisionner un volume FlexGroup. Lorsque l'agrégat est mis à jour au SVM, il est mis à jour automatiquement dans Trident par SVM d'interrogation sans avoir à redémarrer le contrôleur Trident. Lorsque vous avez configuré un agrégat spécifique dans Trident pour provisionner des volumes, si l'agrégat est renommé ou déplacé hors du SVM, le back-end passe à l'état Failed dans Trident lors de l'interrogation de l'agrégat du SVM. Il faut remplacer l'agrégat par un agrégat présent sur la SVM ou le retirer complètement pour remettre le back-end en ligne.	« »

Paramètre	Description	Valeur par défaut
`limitAggregateUsage`	Echec du provisionnement si l'utilisation est supérieure à ce pourcentage. Si vous utilisez un backend Amazon FSX for NetApp ONTAP, ne spécifiez pas `limitAggregateUsage`. Les fournies `fsxadmin` et `vsadmin` ne contiennent pas les autorisations requises pour récupérer l'utilisation des agrégats et la limiter à l'aide de Trident.	« » (non appliqué par défaut)
`limitVolumeSize`	Echec du provisionnement si la taille du volume demandé est supérieure à cette valeur. Limite également la taille maximale des volumes qu'il gère pour 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. Exemple, {"api":false, "method":true} Ne pas utiliser sauf si vous effectuez un dépannage et que vous avez besoin d'un vidage de journal détaillé.	`null`
`useREST`	Paramètre booléen pour utiliser les API REST de ONTAP. `useREST` Lorsqu'il est défini sur `true`, Trident utilise les API REST ONTAP pour communiquer avec le back-end ; lorsqu'il est défini sur `false`, Trident utilise les appels ZAPI ONTAP pour communiquer avec le back-end. Cette fonctionnalité requiert ONTAP 9.11.1 et versions ultérieures. En outre, le rôle de connexion ONTAP utilisé doit avoir accès à l' `ontap` application. Ceci est satisfait par les rôles et prédéfinis `vsadmin` `cluster-admin` . A partir de la version Trident 24.06 et de ONTAP 9.15.1 ou ultérieure, `userREST` est défini sur `true` par défaut ; passez `useREST` à `false` pour utiliser les appels ZAPI ONTAP. `useREST` Est pleinement qualifié pour NVMe/TCP.	`true` Pour ONTAP 9.15.1 ou version ultérieure, sinon `false`.
`sanType`	Utilisez pour sélectionner `iscsi` pour iSCSI, `nvme` pour NVMe/TCP ou `fcp` pour SCSI over Fibre Channel (FC). **'fcp' (SCSI over FC) est une fonctionnalité de présentation technique dans la version Trident 24.10.**	`iscsi` si vide

Paramètre	Description	Valeur par défaut
`formatOptions`	Utilisez `formatOptions` pour spécifier des arguments de ligne de commande pour la `mkfs` commande, qui seront appliqués chaque fois qu'un volume est formaté. Vous pouvez ainsi formater le volume en fonction de vos préférences. Assurez-vous de spécifier les options de formatage similaires à celles des options de commande mkfs, à l'exception du chemin du périphérique. Exemple : « -E nojeter » **Pris en charge pour `ontap-san` les `ontap-san-economy` pilotes et uniquement.**	
`limitVolumePoolSize`	Taille maximale des FlexVol pouvant être demandées lors de l'utilisation de LUN dans le back-end ONTAP-san Economy.	« » (non appliqué par défaut)
`denyNewVolumePools`	Limite les `ontap-san-economy` systèmes back-end à la création de nouveaux volumes FlexVol afin qu'ils contiennent leurs LUN. Seuls les volumes FlexVol préexistants sont utilisés pour provisionner les nouveaux volumes persistants.	

Recommandations pour l'utilisation des options de format

Trident recommande l'option suivante pour accélérer le processus de formatage :

-E nojeter:

- Conservez, n'essayez pas de supprimer des blocs au moment mkfs (la suppression initiale des blocs est utile sur les périphériques SSD et le stockage fragmenté/à provisionnement fin). Ceci remplace l'option obsolète "-K" et s'applique à tous les systèmes de fichiers (xfs, ext3 et ext4).

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 ; « aucun » (fin) ou « volume » (épais)	« aucun »
`snapshotPolicy`	Règle Snapshot à utiliser	« aucun »

Paramètre	Description	Valeur par défaut
`qosPolicy`	QoS policy group à affecter pour les volumes créés. Choisissez une de qosPolicy ou adaptiveQosPolicy par pool de stockage/back-end. L'utilisation de groupes de règles de qualité de service avec Trident nécessite ONTAP 9.8 ou une version ultérieure. Vous devez utiliser un groupe de règles QoS non partagé et vous assurer que le groupe de règles est appliqué à chaque composant individuellement. Un groupe de règles de QoS partagées applique le 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 de volume réservé pour les snapshots	« 0 » si `snapshotPolicy` est « aucun », sinon « »
`splitOnClone`	Séparer un clone de son parent lors de sa création	« faux »
`encryption`	Activez le chiffrement de volume NetApp (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, tout volume provisionné dans Trident est activé. Pour plus d'informations, reportez-vous à la section : "Fonctionnement de Trident avec NVE et NAE".	« faux »
`luksEncryption`	Activez le cryptage LUKS. Reportez-vous à la section "Utiliser la configuration de clé unifiée Linux (LUKS)". Le cryptage LUKS n'est pas pris en charge pour NVMe/TCP.	« »
`securityStyle`	Style de sécurité pour les nouveaux volumes	`unix`
`tieringPolicy`	Règle de hiérarchisation à utiliser « aucun »	« Snapshot uniquement » pour la configuration SVM-DR antérieure à ONTAP 9.5
`nameTemplate`	Modèle pour créer des noms de volume personnalisés.	« »

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
debugTraceFlags:
  api: false
  method: true
defaults:
  spaceReserve: volume
  qosPolicy: standard
  spaceAllocation: 'false'
  snapshotPolicy: default
  snapshotReserve: '10'
```

Pour tous les volumes créés à l'aide du `ontap-san` pilote, Trident ajoute 10 % de capacité supplémentaire au FlexVol pour prendre en charge les métadonnées des LUN. La LUN sera provisionnée avec la taille exacte que l'utilisateur demande dans la demande de volume persistant. Trident ajoute 10 % au FlexVol (s'affiche en tant que 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 par `snapshotReserve`, Trident calcule la taille des volumes comme suit :

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

Le modèle 1.1 représente les 10 % supplémentaires de Trident ajoutés au FlexVol pour prendre en charge les métadonnées de LUN. Pour `snapshotReserve` = 5 % et demande de volume persistant = 5 Gio, la taille totale du volume est de 5,79 Gio et la taille disponible est de 5,5 Gio. `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 on NetApp ONTAP avec Trident, nous vous recommandons de spécifier des noms DNS pour les LIF au lieu d'adresses IP.

Exemple de SAN ONTAP

Il s'agit d'une configuration de base utilisant le `ontap-san` conducteur.

```
---
version: 1
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_iscsi
labels:
  k8scluster: test-cluster-1
  backend: testcluster1-sanbackend
username: vsadmin
password: <password>
```

Exemple d'économie SAN ONTAP

```
---
version: 1
storageDriverName: ontap-san-economy
managementLIF: 10.0.0.1
svm: svm_iscsi_eco
username: vsadmin
password: <password>
```

1. exemple

Vous pouvez configurer le back-end pour éviter d'avoir à mettre à jour manuellement la définition du back-end après le basculement et le rétablissement pendant "Réplication et restauration des SVM".

Pour un basculement et un rétablissement fluides, préciser le SVM en utilisant `managementLIF` et omettre le `dataLIF` et `svm` paramètres. Par exemple :

```
---
version: 1
storageDriverName: ontap-san
managementLIF: 192.168.1.66
username: vsadmin
password: password
```

Exemple d'authentification basée sur un certificat

Dans cet exemple de configuration de base `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
clientCertificate: ZXR0ZXJwYXB...ICMgJ3BhcGVyc2
clientPrivateKey: vciwKIyAgZG...0cnksIGRlc2NyaX
trustedCACertificate: zcyBbaG...b3Igb3duIGNsYXNz
```

Exemples CHAP bidirectionnels

Ces exemples créent un backend avec `useCHAP` réglez sur `true`.

Exemple CHAP de SAN ONTAP

```
---
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
username: vsadmin
password: <password>
```

Exemple CHAP d'économie SAN ONTAP

```
---
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
username: vsadmin
password: <password>
```

Exemple NVMe/TCP

Un SVM doit être configuré avec NVMe sur votre back-end ONTAP. Il s'agit d'une configuration back-end de base pour NVMe/TCP.

```
---
version: 1
backendName: NVMeBackend
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_nvme
username: vsadmin
password: password
sanType: nvme
useREST: true
```

Exemple de configuration back-end avec nomTemplate

```
---
version: 1
storageDriverName: ontap-san
backendName: ontap-san-backend
managementLIF: <ip address>
svm: svm0
username: <admin>
password: <password>
defaults: {
    "nameTemplate":
"{{.volume.Name}}_{{.labels.cluster}}_{{.volume.Namespace}}_{{.volume.R
equestName}}"
},
"labels": {"cluster": "ClusterA", "PVC":
"{{.volume.Namespace}}_{{.volume.RequestName}}"}
```

Exemple de formatoptions pour le pilote </code> <code> ONTAP-san-economy

```
version: 1
storageDriverName: ontap-san-economy
managementLIF: ''
svm: svm1
username: ''
password: "!"
storagePrefix: whelk_
debugTraceFlags:
  method: true
  api: true
defaults:
  formatOptions: "-E nodiscard"
```

Exemples de systèmes back-end avec pools virtuels

Dans ces exemples de fichiers de définition back-end, des valeurs par défaut spécifiques sont définies pour tous les pools de stockage, tels que `spaceReserve` aucune, `spaceAllocation` lors de la fausse idée, et `encryption` faux. Les pools virtuels sont définis dans la section stockage.

Trident définit les étiquettes de provisionnement dans le champ « Commentaires ». Les commentaires sont définis sur le FlexVol. 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 ces exemples, certains pools de stockage sont définis comme étant leurs propres `spaceReserve`, `spaceAllocation`, et `encryption` et certains pools remplacent les valeurs par défaut.

Exemple de SAN ONTAP

```
---
version: 1
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_iscsi
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSd6cNwxyz
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'
```

Exemple d'économie SAN ONTAP

```
---
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
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'
- labels:
    department: legal
    creditpoints: '5000'
  zone: us_east_1c
```

```
  defaults:
    spaceAllocation: 'true'
    encryption: 'false'
```

Exemple NVMe/TCP

```
---
version: 1
storageDriverName: ontap-san
sanType: nvme
managementLIF: 10.0.0.1
svm: nvme_svm
username: vsadmin
password: <password>
useREST: true
defaults:
  spaceAllocation: 'false'
  encryption: 'true'
storage:
- labels:
    app: testApp
    cost: '20'
  defaults:
    spaceAllocation: 'false'
    encryption: 'false'
```

Mappage des systèmes back-end aux classes de stockage

Les définitions de classe de stockage suivantes font référence au Exemples de systèmes back-end avec pools virtuels. À l'aide du `parameters.selector` Chaque classe de stockage indique quels pools virtuels peuvent être utilisés pour héberger un volume. Les aspects définis dans le pool virtuel sélectionné seront définis pour le volume.

- Le `protection-gold` StorageClass est mappé sur le premier pool virtuel du `ontap-san` back-end. Il s'agit du seul pool offrant une protection de niveau Gold.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-gold
provisioner: csi.trident.netapp.io
parameters:
  selector: "protection=gold"
  fsType: "ext4"
```

- Le `protection-not-gold` StorageClass sera mappé au deuxième et au troisième pool virtuel dans `ontap-san` back-end. Ce sont les seuls pools offrant un niveau de protection autre que Gold.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-not-gold
provisioner: csi.trident.netapp.io
parameters:
  selector: "protection!=gold"
  fsType: "ext4"
```

- Le `app-mysqldb` StorageClass sera mappé sur le troisième pool virtuel dans `ontap-san-economy` back-end. Il s'agit du seul pool offrant la configuration du pool de stockage pour l'application de type mysqldb.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: app-mysqldb
provisioner: csi.trident.netapp.io
parameters:
  selector: "app=mysqldb"
  fsType: "ext4"
```

- Le `protection-silver-creditpoints-20k` StorageClass sera mappé sur le second pool virtuel dans `ontap-san` back-end. Il s'agit de la seule piscine offrant une protection de niveau argent et 20000 points de crédit.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-silver-creditpoints-20k
provisioner: csi.trident.netapp.io
parameters:
  selector: "protection=silver; creditpoints=20000"
  fsType: "ext4"
```

- Le `creditpoints-5k` StorageClass sera mappé sur le troisième pool virtuel dans `ontap-san` back-end et le quatrième pool virtuel dans `ontap-san-economy` back-end. Il s'agit des seules offres de pool avec 5000 points de crédit.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: creditpoints-5k
provisioner: csi.trident.netapp.io
parameters:
  selector: "creditpoints=5000"
  fsType: "ext4"
```

- Le `my-test-app-sc` La classe de stockage est mappée sur `testAPP` pool virtuel dans `ontap-san` pilote avec `sanType: nvme`. Il s'agit de la seule offre de piscine `testApp`.

```
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: my-test-app-sc
provisioner: csi.trident.netapp.io
parameters:
  selector: "app=testApp"
  fsType: "ext4"
```

Trident décide du pool virtuel sélectionné et s'assure que les besoins en stockage sont satisfaits.

Pilotes NAS ONTAP

Présentation du pilote NAS ONTAP

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

Détails du pilote NAS ONTAP

Trident fournit les pilotes de stockage NAS suivants pour communiquer avec le cluster ONTAP. Les modes d'accès pris en charge sont : *ReadWriteOnce* (RWO), *ReadOnlyMey* (ROX), *ReadWriteMaly* (RWX), *ReadWriteOncePod* (RWOP).

Conducteur	Protocole	Mode Volume	Modes d'accès pris en charge	Systèmes de fichiers pris en charge
`ontap-nas`	NFS PME	Système de fichiers	RWO, ROX, RWX, RWOP	« », `nfs`, `smb`
`ontap-nas-economy`	NFS PME	Système de fichiers	RWO, ROX, RWX, RWOP	« », `nfs`, `smb`
`ontap-nas-flexgroup`	NFS PME	Système de fichiers	RWO, ROX, RWX, RWOP	« », `nfs`, `smb`

- Utiliser `ontap-san-economy` uniquement si le nombre d'utilisations du volume persistant doit être supérieur à "Limites de volume ONTAP prises en charge".
- Utiliser `ontap-nas-economy` uniquement si le nombre d'utilisations du volume persistant doit être supérieur à "Limites de volume ONTAP prises en charge" 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

Trident s'attend à être exécuté en tant qu'administrateur ONTAP ou SVM, en général avec l'utilisateur du cluster ou un `vsadmin` utilisateur SVM, ou en tant qu' `admin`utilisateur avec un nom différent et le même rôle.

Pour les déploiements Amazon FSX pour NetApp ONTAP, Trident prévoit d'être exécuté en tant qu'administrateur ONTAP ou SVM, en utilisant l'utilisateur du cluster `fsxadmin` ou un `vsadmin` utilisateur SVM, ou un utilisateur avec un nom différent ayant le même rôle. `fsxadmin`L'utilisateur est un remplaçant limité pour l'utilisateur admin du cluster.

Si vous utilisez le `limitAggregateUsage` paramètre, les autorisations d'administration du cluster sont requises. Lors de l'utilisation d'Amazon FSX for NetApp ONTAP avec Trident, le `limitAggregateUsage` paramètre ne fonctionnera pas avec les `vsadmin` comptes d'utilisateur et `fsxadmin`. L'opération de configuration échoue si vous spécifiez ce paramètre.

S'il est possible de créer au sein de ONTAP un rôle plus restrictif 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 les exigences, les options d'authentification et les règles d'exportation pour la configuration d'un back-end ONTAP avec des pilotes NAS ONTAP.

De formation

- Pour tous les systèmes ONTAP back-end, Trident requiert au moins un agrégat attribué à la SVM.
- Vous pouvez 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. Reportez-vous à la section "ici" pour en savoir plus.
- Trident prend en charge les volumes SMB montés sur les pods s'exécutant sur les nœuds Windows uniquement. Voir Préparez-vous au provisionnement des volumes SMB pour plus de détails.

Authentifiez le back-end ONTAP

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

- Basé sur les informations d'identification : ce mode requiert des autorisations suffisantes pour le backend ONTAP. Il est recommandé d'utiliser un compte associé à un rôle de connexion de sécurité prédéfini, par exemple `admin` ou `vsadmin` Pour garantir une compatibilité maximale avec les versions ONTAP.
- Basé sur un certificat : ce mode nécessite l'installation d'un certificat sur le back-end pour que Trident puisse communiquer avec un cluster ONTAP. 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

Trident exige que les identifiants d'un administrateur SVM-scoped/cluster-scoped communiquent avec le back-end ONTAP. Il est recommandé d'utiliser des rôles standard prédéfinis tels que `admin` ou `vsadmin`. La compatibilité avec les futures versions d'ONTAP qui exposent les API de fonctionnalités à utiliser dans les futures versions d'Trident est ainsi garantie. Un rôle de connexion de sécurité personnalisé peut être créé et utilisé avec 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 back-end réussie indique que Trident peut communiquer avec le back-end ONTAP et gérer les futures opérations de volume.

Créez un rôle ONTAP personnalisé pour Trident

Vous pouvez créer un rôle de cluster ONTAP avec une Privileges minimale afin de ne pas avoir à utiliser le rôle ONTAP admin pour effectuer des opérations dans Trident. Lorsque vous incluez le nom d'utilisateur dans une configuration Trident backend, Trident utilise le rôle de cluster ONTAP que vous avez créé pour effectuer les opérations.

Pour plus d'informations sur la création de rôles personnalisés Trident, reportez-vous à la section"Générateur de rôle personnalisé Trident".

Utilisation de l'interface de ligne de commandes ONTAP

1. Créez un rôle à l'aide de la commande suivante :

```
security login role create <role_name\> -cmddirname "command" -access all
–vserver <svm_name\>
```

2. Créez un nom d'utilisateur pour l'utilisateur Trident :

```
security login create -username <user_name\> -application ontapi
-authmethod <password\> -role <name_of_role_in_step_1\> –vserver
<svm_name\> -comment "user_description"
```

3. Mapper le rôle à l'utilisateur :

```
security login modify username <user_name\> –vserver <svm_name\> -role
<role_name\> -application ontapi -application console -authmethod
<password\>
```

À l'aide de System Manager

Dans ONTAP System Manager, effectuez les opérations suivantes :

1. **Créer un rôle personnalisé** :
   a. Pour créer un rôle personnalisé au niveau du cluster, sélectionnez **Cluster > Paramètres**.

      (Ou) pour créer un rôle personnalisé au niveau du SVM, sélectionner **stockage > Storage VM > >> Paramètres > `required SVM` utilisateurs et rôles**.

   b. Sélectionnez l'icône de flèche (→) en regard de **utilisateurs et rôles**.
   c. Sélectionnez **+Ajouter** sous **rôles**.
   d. Définissez les règles du rôle et cliquez sur **Enregistrer**.
2. **Mapper le rôle à l'utilisateur Trident**: + effectuez les étapes suivantes sur la page **utilisateurs et rôles** :
   a. Sélectionnez Ajouter l'icône **+** sous **utilisateurs**.
   b. Sélectionnez le nom d'utilisateur requis et sélectionnez un rôle dans le menu déroulant pour **role**.
   c. Cliquez sur **Enregistrer**.

Pour plus d'informations, reportez-vous aux pages suivantes :

- "Rôles personnalisés pour l'administration de ONTAP" ou "Définissez des rôles personnalisés"
- "Travaillez avec les rôles et les utilisateurs"

Gestion des règles d'exportation NFS

Trident utilise des export policy NFS pour contrôler l'accès aux volumes qu'il provisionne.

Trident propose deux options pour les règles d'export :

- Trident peut gérer la politique d'export de manière dynamique. Dans ce mode de fonctionnement,

l'administrateur du stockage spécifie une liste de blocs CIDR qui représentent des adresses IP recevables. Trident ajoute automatiquement aux règles d'export les adresses IP de nœud applicables comprises dans ces plages au moment de la publication. Sinon, lorsqu'aucun CIDR n'est spécifié, toutes les adresses IP de monodiffusion à périmètre global trouvées sur le nœud auquel le volume est publié seront ajoutées à la export policy.

- Les administrateurs du stockage peuvent créer une export-policy et ajouter des règles manuellement. Trident utilise la export policy par défaut sauf si un autre nom de export policy est spécifié dans la configuration.

Gérez les règles d'exportation de manière dynamique

Trident permet de gérer de manière dynamique les politiques d'exportation des 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 restreindre l'accès au cluster de stockage uniquement aux nœuds workers qui montez des volumes et dont les adresses IP se situent dans la plage spécifiée, et de prendre en charge une gestion automatisée et précise.

> N'utilisez pas NAT (Network Address Translation) lorsque vous utilisez des stratégies d'exportation dynamiques. Avec NAT, le contrôleur de stockage voit l'adresse NAT front-end et non l'adresse IP réelle de l'hôte. L'accès sera donc refusé lorsqu'aucune correspondance n'est trouvée dans les règles d'exportation.

> Dans Trident 24.10, `ontap-nas` le pilote de stockage continuera à fonctionner comme dans les versions précédentes ; aucune modification n'a été apportée au pilote ONTAP-nas. Seul `ontap-nas-economy` le pilote de stockage dispose d'un contrôle d'accès granulaire basé sur les volumes dans Trident 24.10.

Exemple

Deux options de configuration doivent être utilisées. Voici un exemple de définition de back-end :

```
---
version: 1
storageDriverName: ontap-nas-economy
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. Respectez toujours les bonnes pratiques recommandées par NetApp pour dédier une SVM à Trident.

Voici une explication du fonctionnement de cette fonction à l'aide de l'exemple ci-dessus :

- `autoExportPolicy` est défini sur `true`. Cela signifie que Trident crée une export policy pour chaque volume provisionné avec ce back-end pour la `svm1` SVM et gère l'ajout et la suppression de règles à l'aide de `autoexportCIDRs` blocs d'adresse. Tant qu'un volume n'est pas rattaché à un nœud, le volume utilise une export policy vide sans règle pour empêcher tout accès indésirable à ce volume. Lorsqu'un volume est publié sur un nœud, Trident crée une export policy portant le même nom que le qtree sous-jacent contenant l'IP de nœud dans le bloc CIDR spécifié. Ces adresses IP seront également ajoutées à la export policy utilisée par le FlexVol parent.
  - Par exemple :
    - Back-end UUID 403b5326-8482-40db-96d0-d83fb3f4daec
    - `autoExportPolicy` réglez sur `true`
    - préfixe de stockage `trident`
    - UUID de PVC a79bcf5f-7b6d-4a40-9876-e2551f159c1c
    - Qtree nommée Trident_pvc_a79bcf5f_7b6d_4a40_9876_e2551f159c1c crée une export policy pour la FlexVol nommée, une export policy pour le qtree `trident_pvc_a79bcf5f_7b6d_4a40_9876_e2551f159c1c` nommé `trident-403b5326-8482-40db96d0-d83fb3f4daec` et une export policy vide nommée `trident_empty` sur le SVM. Les règles de la FlexVol export policy seront un superset de toutes les règles contenues dans les qtree export policies. Les règles d'export vides seront réutilisées par tous les volumes qui ne sont pas attachés.
- `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, Trident ajoute toutes les adresses de monodiffusion à portée globale trouvées sur les nœuds de travail avec des publications.

Dans cet exemple, l' `192.168.0.0/24`espace d'adresse est fourni. Cela signifie que les adresses IP des nœuds Kubernetes comprises dans cette plage d'adresses avec les publications seront ajoutées à la règle d'export créée par Trident. Lorsque Trident enregistre un nœud sur lequel il s'exécute, il récupère les adresses IP du nœud et les compare aux blocs d'adresse fournis dans `autoExportCIDRs. au moment de la publication, après le filtrage des adresses IP, Trident crée les règles d'export policy pour les adresses IP du client pour le nœud sur lequel il publie.

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.

Une fois que Trident a créé ou mis à jour un back-end, vous pouvez vérifier le back-end à l'aide de `tridentctl` ou du CRD correspondant `tridentbackend` :

```
./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
```

Lorsqu'un nœud est supprimé, Trident vérifie toutes les export policies pour supprimer les règles d'accès correspondant au nœud. En supprimant cette adresse IP de nœud des politiques d'exportation des systèmes back-end gérés, Trident empêche les montages indésirables, sauf si cette adresse IP est réutilisée par un nouveau nœud du cluster.

Pour les systèmes back-end existants, la mise à jour du back-end `tridentctl update backend` permet à Trident de gérer automatiquement les règles d'exportation. Deux nouvelles règles d'exportation nommées en fonction du nom UUID et du nom de qtree du système back-end sont alors créées, le cas échéant. Les volumes présents sur le back-end utiliseront les nouvelles règles d'export créées une fois qu'elles auront été démontées et remontées.

> 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 Trident sur le nœud. Trident mettra ensuite à jour la politique d'exportation des systèmes back-end qu'elle gère pour refléter cette modification de propriété intellectuelle.

Préparez-vous au provisionnement des volumes SMB

Avec un peu de préparation supplémentaire, vous pouvez provisionner des volumes SMB à l'aide de `ontap-nas` pilotes.

> On doit configurer les protocoles NFS et SMB/CIFS sur le SVM pour créer un `ontap-nas-economy` Volume SMB pour ONTAP sur site. La configuration de l'un de ces protocoles entraîne l'échec de la création du volume SMB.

`autoExportPolicy` N'est pas pris en charge pour les volumes SMB.

Avant de commencer

Avant de pouvoir provisionner des volumes SMB, vous devez disposer des éléments suivants :

- Cluster Kubernetes avec un nœud de contrôleur Linux et au moins un nœud worker Windows exécutant Windows Server 2022. Trident prend en charge les volumes SMB montés sur les pods s'exécutant sur les nœuds Windows uniquement.
- Au moins un secret Trident contenant vos informations d'identification 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.

Étapes

1. Pour les ONTAP sur site, vous pouvez créer un partage SMB ou Trident en créer un pour vous.

   Les partages SMB sont requis pour Amazon FSX pour ONTAP.

   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`	Vous pouvez spécifier l'une des options suivantes : le nom d'un partage SMB créé à l'aide de la console de gestion Microsoft ou de l'interface de ligne de commande ONTAP ; un nom permettant à Trident de créer le partage SMB ; ou bien laisser le paramètre vide pour empêcher l'accès au partage commun aux volumes. Ce paramètre est facultatif pour les ONTAP sur site. Ce paramètre est requis pour Amazon FSX pour les systèmes back-end ONTAP et ne peut pas être vide.	`smb-share`
`nasType`	**Doit être défini sur `smb`.** si elle est nulle, la valeur par défaut est `nfs`.	`smb`
`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.**	« »

Options et exemples de configuration du NAS ONTAP

Apprenez à créer et à utiliser des pilotes NAS ONTAP avec votre installation Trident. Cette section fournit des exemples de configuration 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-economy », « ontap-nas-flexgroup », « ontap-san », « ontap-san-economy »
`backendName`	Nom personnalisé ou système back-end de stockage	Nom du pilote + "_" + dataLIF
`managementLIF`	Adresse IP d'un cluster ou LIF de gestion De SVM Un nom de domaine complet (FQDN) peut être spécifié. Peut être configuré pour utiliser des adresses IPv6 si Trident a été installé à l'aide de l'indicateur IPv6. Les adresses IPv6 doivent être définies entre crochets, par exemple `[28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]`. Pour un basculement MetroCluster transparent, consultez le [mcc-best].	« 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 non fourni, Trident récupère les LIFs 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 . Peut être configuré pour utiliser des adresses IPv6 si Trident a été installé à l'aide de l'indicateur IPv6. Les adresses IPv6 doivent être définies entre crochets, par exemple `[28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]`. **Omettre pour MetroCluster.** Voir la [mcc-best].	Adresse spécifiée ou dérivée d'un SVM, si non spécifiée (non recommandé)
`svm`	Serveur virtuel de stockage à utiliser **Omettre pour MetroCluster.** Voir [mcc-best].	Dérivé d'un SVM `managementLIF` est spécifié
`autoExportPolicy`	Activer la création et la mise à jour automatiques des règles d'exportation [booléennes]. Grâce aux `autoExportPolicy` options et `autoExportCIDRs`, Trident peut gérer automatiquement les règles d'export.	faux
`autoExportCIDRs`	Liste des CIDR permettant de filtrer les adresses IP des nœuds Kubernetes par rapport à lorsque `autoExportPolicy` est activé. Grâce aux `autoExportPolicy` options et `autoExportCIDRs`, Trident peut gérer automatiquement les règles d'export.	["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	« »
`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	

Paramètre	Description	Valeur par défaut
`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 Si vous utilisez ONTAP-nas-Economy et un préfixe de stockage de 24 caractères ou plus, le préfixe de stockage n'est pas intégré dans les qtrees, même s'il figure dans le nom du volume.	« trident »
`aggregate`	Agrégat pour le provisionnement (facultatif ; si défini, doit être attribué au SVM) Pour le `ontap-nas-flexgroup` pilote, cette option est ignorée. S'ils ne sont pas affectés, les agrégats disponibles peuvent être utilisés pour provisionner un volume FlexGroup. Lorsque l'agrégat est mis à jour au SVM, il est mis à jour automatiquement dans Trident par SVM d'interrogation sans avoir à redémarrer le contrôleur Trident. Lorsque vous avez configuré un agrégat spécifique dans Trident pour provisionner des volumes, si l'agrégat est renommé ou déplacé hors du SVM, le back-end passe à l'état Failed dans Trident lors de l'interrogation de l'agrégat du SVM. Il faut remplacer l'agrégat par un agrégat présent sur la SVM ou le retirer complètement pour remettre le back-end en ligne.	« »
`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)

Paramètre	Description	Valeur par défaut
FlexgroupAggregateList	Liste des agrégats pour le provisionnement (facultatif ; si défini, doit être affecté au SVM) Tous les agrégats affectés au SVM sont utilisés pour provisionner un volume FlexGroup. Pris en charge pour le pilote de stockage **ONTAP-nas-FlexGroup**. Lorsque la liste des agrégats est mise à jour au SVM, elle est mise à jour automatiquement dans Trident par SVM d'interrogation sans devoir redémarrer le contrôleur Trident. Lorsque vous avez configuré une liste d'agrégats spécifique dans Trident pour provisionner des volumes, si la liste d'agrégats est renommée ou déplacée hors du SVM, le back-end passe à l'état Failed dans Trident lors de l'interrogation de l'agrégat du SVM. Il faut remplacer la liste des agrégats par une liste présente sur la SVM ou la supprimer définitivement pour remettre le système back-end en ligne.	« »
`limitVolumeSize`	Echec du provisionnement si la taille du volume demandé est supérieure à cette valeur. Limite également la taille maximale des volumes gérés pour les qtrees et `qtreesPerFlexvol` permet de personnaliser le nombre maximal de qtrees par FlexVol.	« » (non appliqué par défaut)
`debugTraceFlags`	Indicateurs de débogage à utiliser lors du dépannage. Exemple, {"api":false, "method":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
`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`
`nfsMountOptions`	Liste des options de montage NFS séparée par des virgules. Les options de montage des volumes persistants Kubernetes sont normalement spécifiées dans les classes de stockage, mais si aucune option de montage n'est spécifiée dans une classe de stockage, Trident revient à utiliser les options de montage spécifiées dans le fichier de configuration du back-end de stockage. Si aucune option de montage n'est spécifiée dans la classe de stockage ou le fichier de configuration, Trident ne définit aucune option de montage sur un volume persistant associé.	« »

Paramètre	Description	Valeur par défaut
`qtreesPerFlexvol`	Nombre maximal de qtrees par FlexVol, qui doit être compris dans la plage [50, 300]	« 200 »
`smbShare`	Vous pouvez spécifier l'une des options suivantes : le nom d'un partage SMB créé à l'aide de la console de gestion Microsoft ou de l'interface de ligne de commande ONTAP ; un nom permettant à Trident de créer le partage SMB ; ou bien laisser le paramètre vide pour empêcher l'accès au partage commun aux volumes. Ce paramètre est facultatif pour les ONTAP sur site. Ce paramètre est requis pour Amazon FSX pour les systèmes back-end ONTAP et ne peut pas être vide.	`smb-share`
`useREST`	Paramètre booléen pour utiliser les API REST de ONTAP. `useREST` Lorsqu'il est défini sur `true`, Trident utilise les API REST ONTAP pour communiquer avec le back-end ; lorsqu'il est défini sur `false`, Trident utilise les appels ZAPI ONTAP pour communiquer avec le back-end. Cette fonctionnalité requiert ONTAP 9.11.1 et versions ultérieures. En outre, le rôle de connexion ONTAP utilisé doit avoir accès à l' `ontap` application. Ceci est satisfait par les rôles et prédéfinis `vsadmin` `cluster-admin` . A partir de la version Trident 24.06 et de ONTAP 9.15.1 ou ultérieure, `userREST` est défini sur `true` par défaut ; passez `useREST` à `false` pour utiliser les appels ZAPI ONTAP.	`true` Pour ONTAP 9.15.1 ou version ultérieure, sinon `false`.
`limitVolumePoolSize`	Taille de FlexVol maximale requise lors de l'utilisation de qtrees dans le back-end ONTAP-nas-Economy.	« » (non appliqué par défaut)
`denyNewVolumePools`	Empêche les `ontap-nas-economy` systèmes back-end de créer de nouveaux volumes FlexVol pour contenir leurs qtrees. Seuls les volumes FlexVol préexistants sont utilisés pour provisionner les nouveaux volumes persistants.	

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 qtrees	« vrai »
`spaceReserve`	Mode de réservation d'espace ; « aucun » (fin) ou « volume » (épais)	« aucun »
`snapshotPolicy`	Règle Snapshot à utiliser	« aucun »

Paramètre	Description	Valeur par défaut
`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 de volume réservé pour les snapshots	« 0 » si `snapshotPolicy` est « aucun », sinon « »
`splitOnClone`	Séparer un clone de son parent lors de sa création	« faux »
`encryption`	Activez le chiffrement de volume NetApp (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, tout volume provisionné dans Trident est activé. Pour plus d'informations, reportez-vous à la section : "Fonctionnement de Trident avec NVE et NAE".	« faux »
`tieringPolicy`	Règle de hiérarchisation à utiliser « aucun »	« Snapshot uniquement » pour la configuration SVM-DR antérieure à ONTAP 9.5
`unixPermissions`	Mode pour les nouveaux volumes	« 777 » pour les volumes NFS ; vide (non applicable) pour les volumes SMB
`snapshotDir`	Contrôle l'accès au `.snapshot` répertoire	« True » pour NFSv4 « false » pour NFSv3
`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`.
`nameTemplate`	Modèle pour créer des noms de volume personnalisés.	« »

> L'utilisation de groupes de règles de qualité de service avec Trident nécessite ONTAP 9.8 ou une version ultérieure. Vous devez utiliser un groupe de règles QoS non partagé et vous assurer que le groupe de règles est appliqué à chaque composant individuellement. Un groupe de règles de QoS partagées applique le 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`, Trident utilise maintenant un nouveau calcul pour s'assurer que le FlexVol est correctement dimensionné avec le pourcentage snapshotReserve et le PVC. Lorsque l'utilisateur demande une demande de volume persistant, Trident crée la 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, ce que l'utilisateur a demandé est le volume entier et est un pourcentage de ce volume `snapshotReserve`. Avec Trident 21.07, l'utilisateur demande l'espace inscriptible, et Trident définit le `snapshotReserve` nombre comme le pourcentage du volume dans son ensemble. 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 de 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, une demande de volume persistant de 2 Gio associée à `snapshotReserve=50` la précédente a donné lieu à un volume qui fournit 1 Gio d'espace inscriptible. 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 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.

Exemple d'économie NAS ONTAP

```
---
version: 1
storageDriverName: ontap-nas-economy
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
svm: svm_nfs
username: vsadmin
password: password
```

Exemple de FlexGroup NAS ONTAP

```
---
version: 1
storageDriverName: ontap-nas-flexgroup
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
svm: svm_nfs
username: vsadmin
password: password
```

Exemple MetroCluster

Vous pouvez configurer le back-end pour éviter d'avoir à mettre à jour manuellement la définition du back-end après le basculement et le rétablissement pendant "Réplication et restauration des SVM".

Pour un basculement et un rétablissement fluides, préciser le SVM en utilisant `managementLIF` et omettre le `dataLIF` et `svm` paramètres. Par exemple :

```
---
version: 1
storageDriverName: ontap-nas
managementLIF: 192.168.1.66
username: vsadmin
password: password
```

Exemple de volumes SMB

```
---
version: 1
backendName: ExampleBackend
storageDriverName: ontap-nas
managementLIF: 10.0.0.1
nasType: smb
securityStyle: ntfs
unixPermissions: ""
dataLIF: 10.0.0.2
svm: svm_nfs
username: vsadmin
password: password
```

Exemple d'authentification basée sur un 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
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_
```

Exemple de règle d'export automatique

Cet exemple montre comment vous pouvez demander à Trident d'utiliser des règles d'export dynamiques pour créer et gérer automatiquement les règles d'export. Cela fonctionne de la même manière pour les `ontap-nas-economy` pilotes et `ontap-nas-flexgroup`.

```
---
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
```

Exemple d'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
```

Exemple d'Amazon FSX pour ONTAP avec des volumes SMB

Le `smbShare` Paramètre obligatoire pour FSX for 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
smbShare: smb-share
clientCertificate: ZXR0ZXJwYXB...ICMgJ3BhcGVyc2
clientPrivateKey: vciwKIyAgZG...0cnksIGRlc2NyaX
trustedCACertificate: zcyBbaG...b3Igb3duIGNsYXNz
storagePrefix: myPrefix_
```

Exemple de configuration back-end avec nomTemplate

```
---
version: 1
storageDriverName: ontap-nas
backendName: ontap-nas-backend
managementLIF: <ip address>
svm: svm0
username: <admin>
password: <password>
defaults: {
    "nameTemplate":
"{{.volume.Name}}_{{.labels.cluster}}_{{.volume.Namespace}}_{{.volume.R
equestName}}"
},
"labels": {"cluster": "ClusterA", "PVC":
"{{.volume.Namespace}}_{{.volume.RequestName}}"}
```

Exemples de systèmes back-end avec pools virtuels

Dans les exemples de fichiers de définition back-end présentés ci-dessous, des valeurs par défaut spécifiques sont définies pour tous les pools de stockage, tels que `spaceReserve` aucune, `spaceAllocation` lors de la fausse idée, et `encryption` faux. Les pools virtuels sont définis dans la section stockage.

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`. 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 ces exemples, certains pools de stockage sont définis comme étant leurs propres `spaceReserve`, `spaceAllocation`, et `encryption` et certains pools remplacent les valeurs par défaut.

Exemple de NAS ONTAP

```
---
version: 1
storageDriverName: ontap-nas
managementLIF: 10.0.0.1
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:
    department: legal
    creditpoints: '5000'
  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'
```

Exemple de FlexGroup NAS ONTAP

```
---
version: 1
storageDriverName: ontap-nas-flexgroup
managementLIF: 10.0.0.1
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'
```

Exemple d'économie NAS ONTAP

```
---
version: 1
storageDriverName: ontap-nas-economy
managementLIF: 10.0.0.1
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:
    protection: bronze
    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'
```

Mappage des systèmes back-end aux classes de stockage

Les définitions de classe de stockage suivantes se rapportent à Exemples de systèmes back-end avec pools virtuels. À l'aide du `parameters.selector` Chaque classe de stockage indique quels pools virtuels peuvent être utilisés pour héberger un volume. Les aspects définis dans le pool virtuel sélectionné seront définis pour le volume.

- Le `protection-gold` StorageClass sera mappé au premier et au deuxième pool virtuel de la `ontap-nas-flexgroup` back-end. Il s'agit des seuls pools offrant une protection de niveau Gold.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-gold
provisioner: csi.trident.netapp.io
parameters:
  selector: "protection=gold"
  fsType: "ext4"
```

- Le `protection-not-gold` StorageClass sera mappé au troisième et au quatrième pool virtuel du `ontap-nas-flexgroup` back-end. Ce sont les seuls pools offrant un niveau de protection autre que l'or.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-not-gold
provisioner: csi.trident.netapp.io
parameters:
  selector: "protection!=gold"
  fsType: "ext4"
```

- Le `app-mysqldb` StorageClass sera mappé sur le quatrième pool virtuel du `ontap-nas` back-end. Il s'agit du seul pool offrant la configuration du pool de stockage pour l'application de type mysqldb.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: app-mysqldb
provisioner: csi.trident.netapp.io
parameters:
  selector: "app=mysqldb"
  fsType: "ext4"
```

- TThe `protection-silver-creditpoints-20k` StorageClass sera mappé sur le troisième pool virtuel du `ontap-nas-flexgroup` back-end. Il s'agit de la seule piscine offrant une protection de niveau argent et 20000 points de crédit.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-silver-creditpoints-20k
provisioner: csi.trident.netapp.io
parameters:
  selector: "protection=silver; creditpoints=20000"
  fsType: "ext4"
```

- Le `creditpoints-5k` StorageClass sera mappé sur le troisième pool virtuel du `ontap-nas` back-end et le second pool virtuel dans `ontap-nas-economy` back-end. Il s'agit des seules offres de pool avec 5000 points de crédit.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: creditpoints-5k
provisioner: csi.trident.netapp.io
parameters:
  selector: "creditpoints=5000"
  fsType: "ext4"
```

Trident décide du pool virtuel sélectionné et s'assure que les besoins en stockage sont satisfaits.

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.

Amazon FSX pour NetApp ONTAP

Utilisez 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.

Vous pouvez intégrer votre système de fichiers Amazon FSX pour NetApp ONTAP avec Trident pour vous assurer que les clusters Kubernetes s'exécutant dans Amazon Elastic Kubernetes Service (EKS) peuvent provisionner des volumes persistants de bloc et de fichier soutenus par ONTAP.

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**.

Grâce à Trident avec Amazon FSX pour NetApp ONTAP, vous pouvez vous assurer que les clusters Kubernetes s'exécutant dans Amazon Elastic Kubernetes Service (EKS) peuvent provisionner des volumes persistants de bloc et de fichier soutenus par ONTAP.

De formation

En plus de "Configuration requise pour Trident", pour intégrer FSX for ONTAP avec Trident, vous avez besoin de :

- Un cluster Amazon EKS existant ou un cluster Kubernetes autogéré avec `kubectl` installé.
- Système de fichiers Amazon FSX for NetApp ONTAP et machine virtuelle de stockage (SVM) accessibles 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.

Considérations

- Volumes SMB :
  - Les volumes SMB sont pris en charge à l'aide de `ontap-nas` conducteur uniquement.
  - Les volumes SMB ne sont pas pris en charge par le module d'extension Trident EKS.
  - Trident prend en charge les volumes SMB montés sur les pods s'exécutant sur les nœuds Windows uniquement. Voir "Préparez-vous au provisionnement des volumes SMB" pour plus de détails.
- Avant Trident 24.02, les volumes créés sur les systèmes de fichiers Amazon FSX pour lesquels les sauvegardes automatiques sont activées ne pouvaient pas être supprimés par Trident. Pour éviter ce problème dans Trident 24.02 ou version ultérieure, spécifiez `fsxFilesystemID` , AWS , AWS `apiRegion` `apikey` et AWS `secretKey` dans le fichier de configuration back-end pour AWS FSX pour ONTAP.

> Si vous spécifiez un rôle IAM dans Trident, vous pouvez omettre de spécifier explicitement les `apiRegion` champs , `apiKey` et `secretKey` dans Trident. Pour plus d'informations, reportez-vous "Exemples et options de configuration de FSX pour ONTAP"à .

Authentification

Trident propose deux modes d'authentification.

- Basé sur les informations d'identification (recommandé) : stocke les informations d'identification de manière sécurisée dans AWS secrets Manager. Vous pouvez utiliser `fsxadmin` l'utilisateur pour votre système de fichiers ou l' `vsadmin` utilisateur configuré pour votre SVM.

> Trident s'attend à être exécuté en tant qu' `vsadmin`utilisateur SVM ou en tant qu'utilisateur avec un nom différent qui a le même rôle. Amazon FSX pour NetApp ONTAP a un `fsxadmin` utilisateur qui remplace de façon limitée l'utilisateur du cluster ONTAP `admin`. Nous vous recommandons vivement d'utiliser `vsadmin` Trident.

- Basé sur des certificats : Trident communiquera avec le SVM sur votre système de fichiers FSX à l'aide d'un certificat installé sur votre SVM.

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"

Créez un rôle IAM et un code secret AWS

Vous pouvez configurer les pods Kubernetes pour accéder aux ressources AWS en vous authentifiant en tant que rôle IAM AWS au lieu de fournir des informations d'identification

AWS explicites.

> Pour vous authentifier à l'aide d'un rôle IAM AWS, un cluster Kubernetes doit être déployé à l'aide d'EKS.

Créez un secret AWS Secret Manager

Cet exemple crée un secret AWS Secret Manager pour stocker les informations d'identification Trident CSI :

```
aws secretsmanager create-secret --name trident-secret --description "Trident CSI
credentials" --secret-string "{"user":"vsadmin","password":"<svmpassword>"}"
```

Créer une politique IAM

Les exemples suivants créent une politique IAM à l'aide de l'interface de ligne de commande AWS :

```
aws iam create-policy --policy-name AmazonFSxNCSIDriverPolicy --policy-document
file://policy.json --description "This policy grants access to Trident CSI to
FSxN and Secret manager"
```

Fichier JSON de règles :

```
policy.json:
{
    "Statement": [
        {
            "Action": [
                "fsx:DescribeFileSystems",
                "fsx:DescribeVolumes",
                "fsx:CreateVolume",
                "fsx:RestoreVolumeFromSnapshot",
                "fsx:DescribeStorageVirtualMachines",
                "fsx:UntagResource",
                "fsx:UpdateVolume",
                "fsx:TagResource",
                "fsx:DeleteVolume"
            ],
            "Effect": "Allow",
            "Resource": "*"
        },
        {
            "Action": "secretsmanager:GetSecretValue",
            "Effect": "Allow",
            "Resource": "arn:aws:secretsmanager:<aws-region>:<aws-account-
id>:secret:<aws-secret-manager-name>"
        }
    ],
    "Version": "2012-10-17"
}
```

Créer un rôle IAM pour le compte de service

L'exemple suivant crée un rôle IAM pour le compte de service dans EKS :

```
eksctl create iamserviceaccount --name trident-controller --namespace trident
--cluster <my-cluster> --role-name <AmazonEKS_FSxN_CSI_DriverRole> --role-only
--attach-policy-arn arn:aws:iam::aws:policy/service-
role/AmazonFSxNCSIDriverPolicy --approve
```

Installer Astra Trident

ASTRA Trident simplifie la gestion du stockage Amazon FSX pour NetApp ONTAP dans Kubernetes pour que vos développeurs et administrateurs puissent donner la priorité au déploiement d'applications.

Vous pouvez installer Astra Trident à l'aide de l'une des méthodes suivantes :

- Gouvernail

- Module complémentaire EKS

```
If you want to make use of the snapshot functionality, install the CSI
snapshot controller add-on. Refer to
https://docs.aws.amazon.com/eks/latest/userguide/csi-snapshot-
controller.html.
```

Installez Astra Trident via Helm

1. Téléchargez le package d'installation d'Astra 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 d'Astra Trident à partir de la section des ressources sur GitHub.
   ```
   wget https://github.com/NetApp/trident/releases/download/v24.10.0/trident-
   installer-24.10.0.tar.gz
   tar -xf trident-installer-24.10.0.tar.gz
   cd trident-installer
   ```

2. Définissez les valeurs des indicateurs **cloud Provider** et **cloud Identity** à l'aide des variables d'environnement suivantes :

   ```
   export CP="AWS"
   export CI="'eks.amazonaws.com/role-arn:
   arn:aws:iam::<accountID>:role/<AmazonEKS_FSxN_CSI_DriverRole>'"
   ```

   L'exemple suivant installe Astra Trident et définit le `cloud-provider` drapeau sur `$CP`, et `cloud-identity` sur `$CI`:

   ```
   helm install trident trident-operator-100.2410.0.tgz --set
   cloudProvider=$CP --set cloudIdentity=$CI --namespace trident
   ```

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

   ```
   helm list -n trident
   ```

   ```
   NAME                NAMESPACE   REVISION    UPDATED
   STATUS       CHART                           APP VERSION

   trident-operator    trident     1           2024-10-14 14:31:22.463122
   +0300 IDT    deployed     trident-operator-100.2410.0    24.10.0
   ```

Installez Astra Trident via le module complémentaire EKS

Le module complémentaire Astra Trident EKS inclut les derniers correctifs de sécurité et de bogues, et il est validé par AWS pour fonctionner avec Amazon EKS. Le module complémentaire EKS vous permet de vous assurer de manière cohérente que vos clusters Amazon EKS sont sécurisés et stables et de réduire la quantité de travail à effectuer pour installer, configurer et mettre à jour des modules complémentaires.

Prérequis

Vérifiez les points suivants avant de configurer le module complémentaire Astra Trident pour AWS EKS :

- Un compte de cluster Amazon EKS avec abonnement complémentaire
- Autorisations AWS sur AWS Marketplace :
  ```
  "aws-marketplace:ViewSubscriptions",
  "aws-marketplace:Subscribe",
  "aws-marketplace:Unsubscribe
  ```
- Type ami : Amazon Linux 2 (AL2_x86_64) ou Amazon Linux 2 Arm (AL2_ARM_64)
- Type de nœud : AMD ou ARM
- Un système de fichiers Amazon FSX pour NetApp ONTAP

Activez le module complémentaire Astra Trident pour AWS

Groupe EKS

Les exemples de commandes suivants installent le module complémentaire Astra Trident EKS :

```
eksctl create addon --cluster clusterName --name netapp_trident-operator
--version v24.6.1-eksbuild
eksctl create addon --cluster clusterName --name netapp_trident-operator
--version v24.6.1-eksbuild.1
```
(avec une version dédiée)

> Lorsque vous configurez le paramètre facultatif `cloudIdentity` , assurez-vous de spécifier `cloudProvider` lors de l'installation de Trident à l'aide du module complémentaire EKS.

Console de gestion

1. Ouvrez la console Amazon EKS à l'adresse https://console.aws.amazon.com/eks/home#/clusters.
2. Dans le volet de navigation de gauche, cliquez sur **clusters**.
3. Cliquez sur le nom du cluster pour lequel vous souhaitez configurer le module complémentaire NetApp Trident CSI.
4. Cliquez sur **Compléments**, puis cliquez sur **obtenir plus de modules complémentaires**.
5. Sur la page **S*elect add-ons**, procédez comme suit :
   a. Dans la section EKS-addons d'AWS Marketplace, cochez la case **Astra Trident by NetApp**.
   b. Cliquez sur **Suivant**.
6. Sur la page **configurer les compléments sélectionnés**, procédez comme suit :
   a. Sélectionnez la **version** que vous souhaitez utiliser.
   b. Pour **Sélectionner le rôle IAM**, laisser à **non défini**.
   c. Développez **Paramètres de configuration facultatifs**, suivez le schéma de configuration **Compléments** et définissez le paramètre configurationValues dans la section **valeurs de configuration** sur le fil de rôle que vous avez créé à l'étape précédente (la valeur doit être au format suivant : `eks.amazonaws.com/role-arn: arn:aws:iam::464262061435:role/AmazonEKS_FSXN_CSI_DriverRole`). Si vous sélectionnez remplacer pour la méthode de résolution des conflits, un ou plusieurs des paramètres du module complémentaire existant peuvent être remplacés par les paramètres du module complémentaire Amazon EKS. Si vous n'activez pas cette option et qu'il y a un conflit avec vos paramètres existants, l'opération échoue. Vous pouvez utiliser le message d'erreur qui en résulte pour résoudre le conflit. Avant de sélectionner cette option, assurez-vous que le module complémentaire Amazon EKS ne gère pas les paramètres que vous devez gérer vous-même.

   > Lorsque vous configurez le paramètre facultatif `cloudIdentity` , assurez-vous de spécifier `cloudProvider` lors de l'installation de Trident à l'aide du module complémentaire EKS.

7. Choisissez **Suivant**.
8. Sur la page **consulter et ajouter**, choisissez **Créer**.

   Une fois l'installation du module complémentaire terminée, le module complémentaire installé s'affiche.

CLI AWS

1. Créez le `add-on.json` fichier :

```
add-on.json
{

   "clusterName": "<eks-cluster>",
   "addonName": "netapp_trident-operator",
   "addonVersion": "v24.6.1-eksbuild.1",
   "serviceAccountRoleArn": "arn:aws:iam::123456:role/astratrident-
role",
   "configurationValues": "{"cloudIdentity":
"'eks.amazonaws.com/role-arn: arn:aws:iam::123456:role/astratrident-
role'",
   "cloudProvider": "AWS"}"
}
```

> Lorsque vous configurez le paramètre facultatif `cloudIdentity` , assurez-vous que vous spécifiez `AWS` en tant que `cloudProvider` lors de l'installation de Trident à l'aide du module complémentaire EKS.

2. Installer le module complémentaire Astra Trident EKS »

```
aws eks create-addon --cli-input-json file://add-on.json
```

Mettez à jour le module complémentaire Astra Trident EKS

Groupe EKS

- Vérifiez la version actuelle de votre module complémentaire FSxN Trident CSI. Remplacez `my-cluster` par le nom de votre cluster.
  `eksctl get addon --name netapp_trident-operator --cluster my-cluster`

  Exemple de sortie :

```
NAME                          VERSION              STATUS    ISSUES
IAMROLE    UPDATE AVAILABLE    CONFIGURATION VALUES
netapp_trident-operator    v24.6.1-eksbuild.1    ACTIVE    0
{"cloudIdentity":"'eks.amazonaws.com/role-arn:
arn:aws:iam::139763910815:role/AmazonEKS_FSXN_CSI_DriverRole'"}
```

- Mettez à jour le complément à la version renvoyée sous MISE À JOUR DISPONIBLE dans la sortie de l'étape précédente.
  `eksctl update addon --name netapp_trident-operator --version v24.6.1-eksbuild.1 --cluster my-cluster --force`

  Si vous supprimez l' `--force` option et que l'un des paramètres du module complémentaire Amazon EKS entre en conflit avec vos paramètres existants, la mise à jour du module complémentaire Amazon EKS échoue ; un message d'erreur s'affiche pour vous aider à résoudre le conflit. Avant de spécifier cette option, assurez-vous que le module complémentaire Amazon EKS ne gère pas les paramètres que vous devez gérer, car ces paramètres sont remplacés par cette option. Pour plus d'informations sur les autres options de ce paramètre, reportez-vous à la section "Addons". Pour plus d'informations sur la gestion de terrain Amazon EKS Kubernetes, reportez-vous à la section "Gestion de terrain Kubernetes".

Console de gestion

1. Ouvrez la console Amazon EKS https://console.aws.amazon.com/eks/home#/clusters.
2. Dans le volet de navigation de gauche, cliquez sur **clusters**.
3. Cliquez sur le nom du cluster pour lequel vous souhaitez mettre à jour le module complémentaire NetApp Trident CSI.
4. Cliquez sur l'onglet **Compléments**.
5. Cliquez sur **Astra Trident by NetApp**, puis sur **Modifier**.
6. Sur la page **configurer Astra Trident par NetApp**, procédez comme suit :
   a. Sélectionnez la **version** que vous souhaitez utiliser.
   b. (Facultatif) vous pouvez développer les **Paramètres de configuration facultatifs** et les modifier si nécessaire.
   c. Cliquez sur **Enregistrer les modifications**.

CLI AWS

L'exemple suivant met à jour le module complémentaire EKS :

```
aws eks update-addon --cluster-name my-cluster netapp_trident-operator vpc-cni
--addon-version v24.6.1-eksbuild.1 \
```

```
--service-account-role-arn arn:aws:iam::111122223333:role/role-name
--configuration-values '{}' --resolve-conflicts --preserve
```

Désinstallez/supprimez le module complémentaire Astra Trident EKS

Vous avez deux options pour supprimer un module complémentaire Amazon EKS :

- **Préserver le logiciel complémentaire sur votre cluster** – cette option supprime la gestion Amazon EKS de tous les paramètres. Il supprime également la possibilité pour Amazon EKS de vous informer des mises à jour et de mettre à jour automatiquement le module complémentaire Amazon EKS après avoir lancé une mise à jour. Cependant, il conserve le logiciel complémentaire sur votre cluster. Cette option fait du complément une installation auto-gérée, plutôt qu'un module complémentaire Amazon EKS. Avec cette option, vous n'avez plus à subir de temps d'indisponibilité. Conservez `--preserve` l'option dans la commande pour conserver le complément.
- **Supprimer entièrement le logiciel complémentaire de votre cluster** – nous vous recommandons de supprimer le module complémentaire Amazon EKS de votre cluster uniquement s'il n'y a pas de ressources qui en dépendent sur votre cluster. Supprimez l' `--preserve` option de la `delete` commande pour supprimer le complément.

Si le complément est associé à un compte IAM, le compte IAM n'est pas supprimé.

Groupe EKS

La commande suivante désinstalle le module complémentaire Astra Trident EKS :
`eksctl delete addon --cluster K8s-arm --name netapp_trident-operator`

Console de gestion

1. Ouvrez la console Amazon EKS à l'adresse https://console.aws.amazon.com/eks/home#/clusters.
2. Dans le volet de navigation de gauche, cliquez sur **clusters**.
3. Cliquez sur le nom du cluster pour lequel vous souhaitez supprimer le module complémentaire NetApp Trident CSI.
4. Cliquez sur l'onglet **Compléments**, puis cliquez sur **Astra Trident by NetApp**.*
5. Cliquez sur **Supprimer**.
6. Dans la boîte de dialogue **Remove netapp_trident-operator confirmation**, procédez comme suit :
   a. Si vous souhaitez qu'Amazon EKS cesse de gérer les paramètres du module complémentaire, sélectionnez **préserver sur le cluster**. Procédez ainsi si vous souhaitez conserver l'extension logicielle sur votre cluster afin de pouvoir gérer tous les paramètres du module complémentaire vous-même.
   b. Entrez **netapp_trident-operator**.
   c. Cliquez sur **Supprimer**.

CLI AWS

Remplacez `my-cluster` par le nom de votre cluster, puis exécutez la commande suivante.

```
aws eks delete-addon --cluster-name my-cluster --addon-name netapp_trident-
operator --preserve
```

Configurez le back-end de stockage

Intégration des pilotes SAN et NAS de ONTAP

Vous pouvez créer un fichier backend en utilisant les informations d'identification du SVM (nom d'utilisateur et mot de passe) stockées dans AWS Secret Manager, comme illustré ci-dessous :

YAML

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-nas
spec:
  version: 1
  storageDriverName: ontap-nas
  backendName: tbc-ontap-nas
  svm: svm-name
  aws:
    fsxFilesystemID: fs-xxxxxxxxxx
  credentials:
    name: "arn:aws:secretsmanager:us-west-2:xxxxxxxx:secret:secret-
name"
    type: awsarn
```

JSON

```
{
  "apiVersion": "trident.netapp.io/v1",
  "kind": "TridentBackendConfig",
  "metadata": {
    "name": "backend-tbc-ontap-nas"
  },
  "spec": {
    "version": 1,
    "storageDriverName": "ontap-nas",
    "backendName": "tbc-ontap-nas",
    "svm": "svm-name",
    "aws": {
      "fsxFilesystemID": "fs-xxxxxxxxxx"
    },
    "managementLIF": null,
    "credentials": {
      "name": "arn:aws:secretsmanager:us-west-2:xxxxxxxx:secret:secret-
name",
      "type": "awsarn"
    }
  }
}
```

Pour plus d’informations sur la création de systèmes back-end, consultez les pages suivantes :

- "Configurer un système back-end avec les pilotes NAS ONTAP"
- "Configurer un système back-end avec les pilotes SAN ONTAP"

Détails du pilote FSX pour ONTAP

Vous pouvez intégrer Trident avec Amazon FSX for 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. Recommandé pour le stockage en mode bloc.
- `ontap-nas`: Chaque volume persistant provisionné est un volume Amazon FSX pour NetApp ONTAP complet. Recommandé pour les protocoles NFS et SMB.
- `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-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 NAS" et "Pilotes SAN".

Exemples de configurations

Configuration pour AWS FSX pour ONTAP avec le gestionnaire de code secret

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-nas
spec:
  version: 1
  storageDriverName: ontap-nas
  backendName: tbc-ontap-nas
  svm: svm-name
  aws:
    fsxFilesystemID: fs-xxxxxxxxxx
  managementLIF:
  credentials:
    name: "arn:aws:secretsmanager:us-west-2:xxxxxxxx:secret:secret-
name"
    type: awsarn
```

Configuration de la classe de stockage 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. 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"
```

Configuration avancée back-end et exemples

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-economy`, `ontap-nas-flexgroup`, `ontap-san`, `ontap-san-economy`
`backendName`	Nom personnalisé ou système back-end de stockage	Nom du pilote + "_" + dataLIF

Paramètre	Description	Exemple
managementLIF	Adresse IP d'un cluster ou LIF de gestion De SVM Un nom de domaine complet (FQDN) peut être spécifié. Peut être configuré pour utiliser des adresses IPv6 si Trident a été installé à l'aide de l'indicateur IPv6. Les adresses IPv6 doivent être définies entre crochets, telles que [28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]. Si vous fournissez fsxFilesystemID sous le aws champ, il n'est pas nécessaire de fournir le managementLIF car Trident récupère les informations du SVM managementLIF auprès d'AWS. Donc, vous devez fournir des informations d'identification pour un utilisateur sous la SVM (par exemple : vsadmin) et l'utilisateur doit avoir le vsadmin rôle.	« 10.0.0.1 », « [2001:1234:abcd::fefe] »
dataLIF	Adresse IP de la LIF de protocole. **Pilotes NAS ONTAP**: Nous vous recommandons de spécifier dataLIF. Si non fourni, Trident récupère les LIFs 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 . **Pilotes SAN ONTAP** : ne pas spécifier pour iSCSI. Trident utilise ONTAP Selective LUN Map pour découvrir les LIF iSCI nécessaires à l'établissement d'une session à chemins multiples. Un avertissement est généré si dataLIF est explicitement défini. Peut être configuré pour utiliser des adresses IPv6 si Trident a été installé à l'aide de l'indicateur IPv6. Les adresses IPv6 doivent être définies entre crochets, telles que [28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555].	

Paramètre	Description	Exemple
`autoExportPolicy`	Activer la création et la mise à jour automatiques des règles d'exportation [booléennes]. Grâce aux `autoExportPolicy` options et `autoExportCIDRs`, Trident peut gérer automatiquement les règles d'export.	`false`
`autoExportCIDRs`	Liste des CIDR permettant de filtrer les adresses IP des nœuds Kubernetes par rapport à lorsque `autoExportPolicy` est activé. Grâce aux `autoExportPolicy` options et `autoExportCIDRs`, Trident peut gérer automatiquement les règles d'export.	« [« 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	« »
`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.
`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`

Paramètre	Description	Exemple
`limitAggregateUsage`	**Ne spécifiez pas pour Amazon FSX pour NetApp ONTAP.** Les fournies `fsxadmin` et `vsadmin` ne contiennent pas les autorisations requises pour récupérer l'utilisation des agrégats et la limiter à l'aide de 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)
`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 persistants Kubernetes sont normalement spécifiées dans les classes de stockage, mais si aucune option de montage n'est spécifiée dans une classe de stockage, Trident revient à utiliser les options de montage spécifiées dans le fichier de configuration du back-end de stockage. Si aucune option de montage n'est spécifiée dans la classe de stockage ou le fichier de configuration, Trident ne définit aucune option de montage sur un volume persistant associé.	« »

Paramètre	Description	Exemple
`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`	Vous pouvez spécifier l'une des options suivantes : le nom d'un partage SMB créé à l'aide de la console de gestion Microsoft ou de l'interface de ligne de commande ONTAP, ou un nom permettant à Trident de créer le partage SMB. Ce paramètre est requis pour Amazon FSX pour les systèmes back-end ONTAP.	`smb-share`
`useREST`	Paramètre booléen pour utiliser les API REST de ONTAP. **Tech Preview** `useREST` est fourni sous forme de **aperçu technique** recommandé pour les environnements de test et non pour les charges de travail de production. Lorsqu'il est défini sur `true`, Trident utilise les API REST ONTAP pour communiquer avec le back-end. Cette fonctionnalité requiert ONTAP 9.11.1 et versions ultérieures. En outre, le rôle de connexion ONTAP utilisé doit avoir accès à l' `ontap` application. Ceci est satisfait par les rôles et prédéfinis `vsadmin` `cluster-admin` .	`false`
`aws`	Vous pouvez spécifier ce qui suit dans le fichier de configuration d'AWS FSX pour ONTAP : - `fsxFilesystemID`: Spécifiez l'ID du système de fichiers AWS FSX. - `apiRegion`: Nom de la région de l'API AWS. - `apikey`: Clé d'API AWS. - `secretKey`: Clé secrète AWS.	 `""` `""` `""`

Paramètre	Description	Exemple
`credentials`	Spécifiez les informations d'identification du SVM FSX à stocker dans AWS Secret Manager. - `name`: Amazon Resource Name (ARN) du secret, qui contient les informations d'identification de SVM. - `type`: Défini sur `awsarn`. Reportez-vous à la section "Créez un secret AWS secrets Manager" pour en savoir plus.	

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	`true`
`spaceReserve`	Mode de réservation d'espace ; "none" (fin) ou "volume" (épais)	`none`
`snapshotPolicy`	Règle Snapshot à utiliser	`none`
`qosPolicy`	QoS policy group à affecter pour les volumes créés. Choisissez une de qosPolicy ou adaptiveQosPolicy par pool de stockage ou back-end. L'utilisation de groupes de règles de qualité de service avec Trident nécessite ONTAP 9.8 ou une version ultérieure. Vous devez utiliser un groupe de règles QoS non partagé et vous assurer que le groupe de règles est appliqué à chaque composant individuellement. Un groupe de règles de QoS partagées applique le 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 `none`, `else` « »

Paramètre	Description	Valeur par défaut
`splitOnClone`	Séparer un clone de son parent lors de sa création	`false`
`encryption`	Activez le chiffrement de volume NetApp (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, tout volume provisionné dans Trident est activé. Pour plus d'informations, reportez-vous à la section : "Fonctionnement de Trident avec NVE et NAE".	`false`
`luksEncryption`	Activez le cryptage LUKS. Reportez-vous à la section "Utiliser la configuration de clé unifiée Linux (LUKS)". SAN uniquement.	« »
`tieringPolicy`	Règle de hiérarchisation à utiliser `none`	`snapshot-only` Pour la configuration SVM-DR antérieure à 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`.

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.

Avant de commencer

Avant de pouvoir provisionner des volumes SMB à l'aide de `ontap-nas` pilote, vous devez avoir les éléments suivants.

- Cluster Kubernetes avec un nœud de contrôleur Linux et au moins un nœud worker Windows exécutant Windows Server 2019. Trident prend en charge les volumes SMB montés sur les pods s'exécutant sur les nœuds Windows uniquement.
- Au moins un secret Trident contenant vos informations d'identification 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.

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`	Vous pouvez spécifier l'une des options suivantes : le nom d'un partage SMB créé à l'aide de la console de gestion Microsoft ou de l'interface de ligne de commande ONTAP, ou un nom permettant à Trident de créer le partage SMB. Ce paramètre est requis pour Amazon FSX pour les systèmes back-end ONTAP.	`smb-share`
`nasType`	**Doit être défini sur `smb`.** si elle est nulle, la valeur par défaut est `nfs`.	`smb`
`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

Paramètre	Description	Exemple
`unixPermissions`	Mode pour les nouveaux volumes. **Doit rester vide pour les volumes SMB.**	« »

Configurez une classe de stockage et un PVC

Configurez un objet StorageClass Kubernetes et créez la classe de stockage pour indiquer à Trident comment provisionner les volumes. Créez un volume persistant et une demande de volume persistant qui utilisent la classe de stockage Kubernetes configurée pour demander l'accès au volume persistant. Vous pouvez ensuite monter le volume persistant sur un pod.

Créer une classe de stockage

Configuration d'un objet StorageClass Kubernetes

Le système "Objet classe de stockage Kubernetes" identifie Trident en tant que mécanisme de provisionnement utilisé pour cette classe indique à Trident comment provisionner un volume. Par exemple :

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-gold
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-nas"
  media: "ssd"
  provisioningType: "thin"
  snapshots: "true"
```

Reportez-vous "Kubernetes et objets Trident"à pour plus de détails sur l'interaction des classes de stockage avec les `PersistentVolumeClaim` paramètres et pour le contrôle de la manière dont Trident provisionne les volumes.

Créer une classe de stockage

Étapes

1. Il s'agit d'un objet Kubernetes, alors utilisez-le `kubectl` Pour la créer dans Kubernetes.

```
kubectl create -f storage-class-ontapnas.yaml
```

2. Vous devriez maintenant voir une classe de stockage **Basic-csi** dans Kubernetes et Trident, et Trident aurait dû détecter les pools sur le back-end.

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

Créer le volume persistant et la demande de volume persistant

A "*Volume persistant*" (PV) est une ressource de stockage physique provisionnée par l'administrateur du cluster sur un cluster Kubernetes. Le "*PersistentVolumeClaim*" (PVC) est une demande d'accès au volume persistant sur le cluster.

Le PVC peut être configuré pour demander un stockage d'une certaine taille ou d'un certain mode d'accès. À l'aide de la classe de stockage associée, l'administrateur du cluster peut contrôler plus que la taille du volume persistant et le mode d'accès, tels que les performances ou le niveau de service.

Après avoir créé le volume persistant et la demande de volume persistant, vous pouvez monter le volume dans un pod.

Exemples de manifestes

Exemple de manifeste de volume persistant

Cet exemple de manifeste montre un volume persistant de base de 10Gi associé à StorageClass `basic-csi`.

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-storage
  labels:
    type: local
spec:
  storageClassName: basic-csi
  capacity:
    storage: 10Gi
  accessModes:
    - ReadWriteMany
  hostPath:
    path: "/my/host/path"
```

Exemples de manifestes de demande de volume persistant

Ces exemples présentent les options de configuration de base de la PVC.

PVC avec accès RWO

Cet exemple montre une demande de volume persistant de base avec accès RWX associée à une classe de stockage nommée `basic-csi`.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc-storage
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: basic-csi
```

PVC avec NVMe/TCP

Cet exemple présente une demande de volume persistant de base pour NVMe/TCP avec accès RWO associée à une classe de stockage nommée `protection-gold`.

```
---
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
name: pvc-san-nvme
spec:
accessModes:
  - ReadWriteOnce
resources:
  requests:
    storage: 300Mi
storageClassName: protection-gold
```

Créer le volume persistant et la demande de volume persistant

Étapes

1. Créer la PV.

```
kubectl create -f pv.yaml
```

2. Vérifiez l'état du PV.

```
kubectl get pv
NAME         CAPACITY  ACCESS MODES  RECLAIM POLICY  STATUS    CLAIM
STORAGECLASS  REASON  AGE
pv-storage  4Gi       RWO           Retain          Available
7s
```

3. Créer la PVC.

```
kubectl create -f pvc.yaml
```

4. Vérifiez l'état de la demande de volume persistant.

```
kubectl get pvc
NAME         STATUS VOLUME     CAPACITY ACCESS MODES STORAGECLASS AGE
pvc-storage Bound  pv-name 2Gi      RWO                          5m
```

Reportez-vous "Kubernetes et objets Trident"à pour plus de détails sur l'interaction des classes de stockage avec les `PersistentVolumeClaim` paramètres et pour le contrôle de la manière dont Trident provisionne les volumes.

Attributs Trident

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

Attribut	Type	Valeurs	Offre	Demande	Pris en charge par
support[1]	chaîne	hdd, hybride, ssd	Le pool contient des supports de ce type ; hybride signifie les deux	Type de support spécifié	ontap-nas, ontap-nas-économie, ontap-nas-flexgroup, ontap-san, solidfire-san
Type de provisionnement	chaîne	fin, épais	Le pool prend en charge cette méthode de provisionnement	Méthode de provisionnement spécifiée	thick : tous les systèmes ONTAP ; thin : tous les systèmes ONTAP et solidfire-san

Attribut	Type	Valeurs	Offre	Demande	Pris en charge par
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

Déploiement de l'application exemple

Déploiement de l'application exemple.

Étapes

1. Montez le volume dans un pod.

```
kubectl create -f pv-pod.yaml
```

Ces exemples montrent les configurations de base pour attacher le PVC à un pod : **Configuration de base** :

```
kind: Pod
apiVersion: v1
metadata:
  name: pv-pod
spec:
  volumes:
    - name: pv-storage
      persistentVolumeClaim:
       claimName: basic
  containers:
    - name: pv-container
      image: nginx
      ports:
        - containerPort: 80
          name: "http-server"
      volumeMounts:
        - mountPath: "/my/mount/path"
          name: pv-storage
```

Vous pouvez surveiller la progression à l'aide de `kubectl get pod --watch`.

2. Vérifiez que le volume est monté sur `/my/mount/path`.

```
kubectl exec -it task-pv-pod -- df -h /my/mount/path
```

```
Filesystem                                                      Size
Used Avail Use% Mounted on
192.168.188.78:/trident_pvc_ae45ed05_3ace_4e7c_9080_d2a83ae03d06  1.1G
320K  1.0G   1% /my/mount/path
```

1. Vous pouvez maintenant supprimer le Pod. L'application Pod n'existera plus, mais le volume restera.

```
kubectl delete pod task-pv-pod
```

Configurez le module d'extension Astra Trident EKS sur un cluster EKS

ASTRA Trident simplifie la gestion du stockage Amazon FSX pour NetApp ONTAP dans Kubernetes pour que vos développeurs et administrateurs puissent donner la priorité au déploiement d'applications. Le module complémentaire Astra Trident EKS inclut les derniers correctifs de sécurité et de bogues, et il est validé par AWS pour fonctionner avec Amazon EKS. Le module complémentaire EKS vous permet de vous assurer de

manière cohérente que vos clusters Amazon EKS sont sécurisés et stables et de réduire la quantité de travail à effectuer pour installer, configurer et mettre à jour des modules complémentaires.

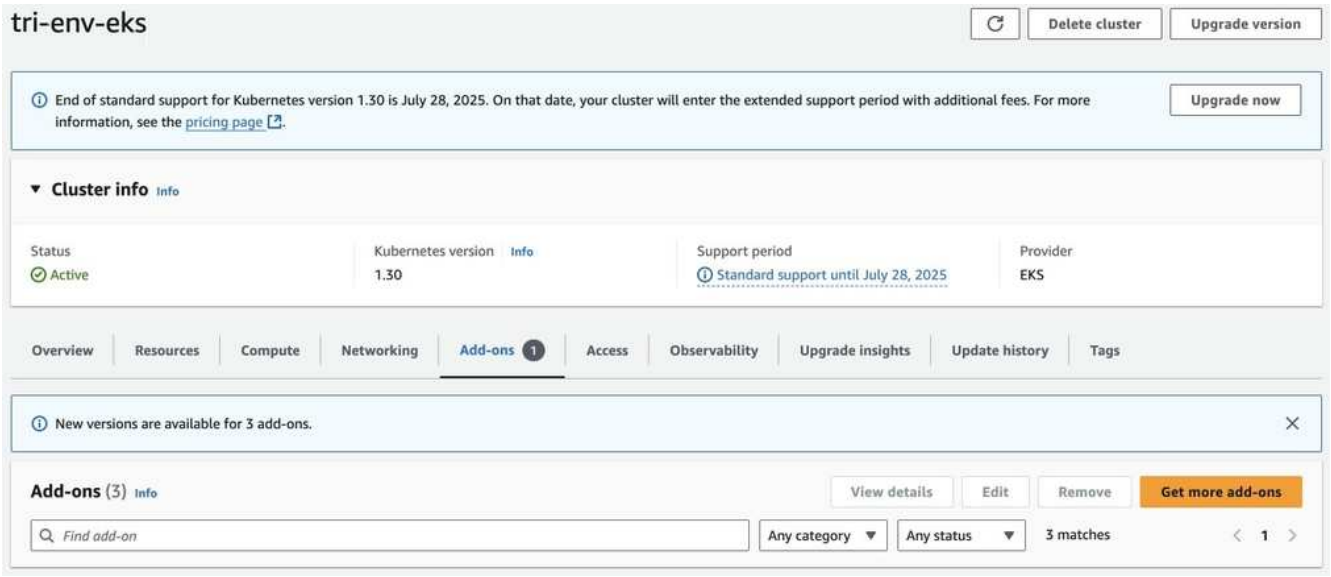
Prérequis

Vérifiez les points suivants avant de configurer le module complémentaire Astra Trident pour AWS EKS :

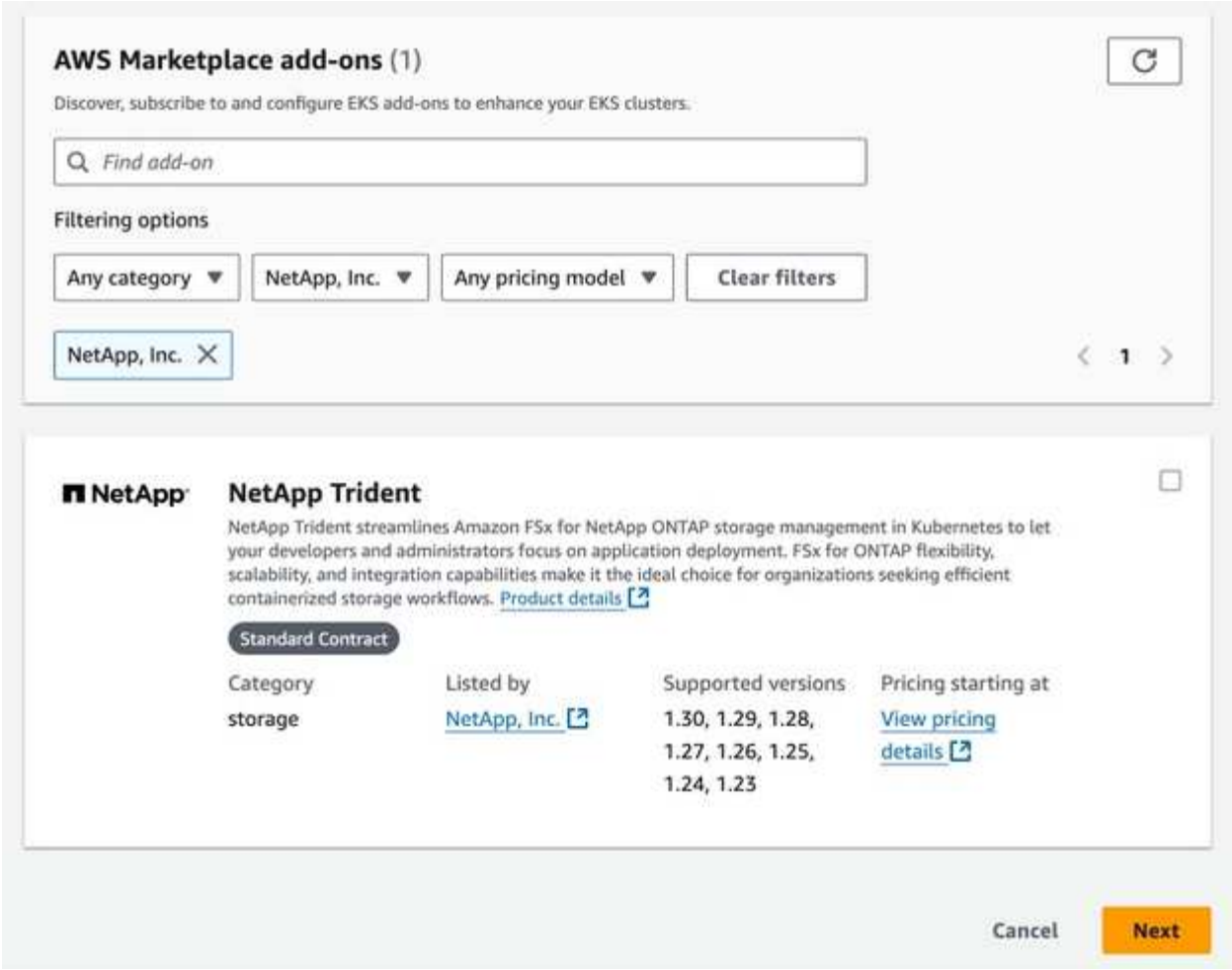
- Un compte de cluster Amazon EKS avec abonnement complémentaire
- Autorisations AWS sur AWS Marketplace :
  `"aws-marketplace:ViewSubscriptions",`
  `"aws-marketplace:Subscribe",`
  `"aws-marketplace:Unsubscribe`
- Type ami : Amazon Linux 2 (AL2_x86_64) ou Amazon Linux 2 Arm (AL2_ARM_64)
- Type de nœud : AMD ou ARM
- Un système de fichiers Amazon FSX pour NetApp ONTAP

Étapes

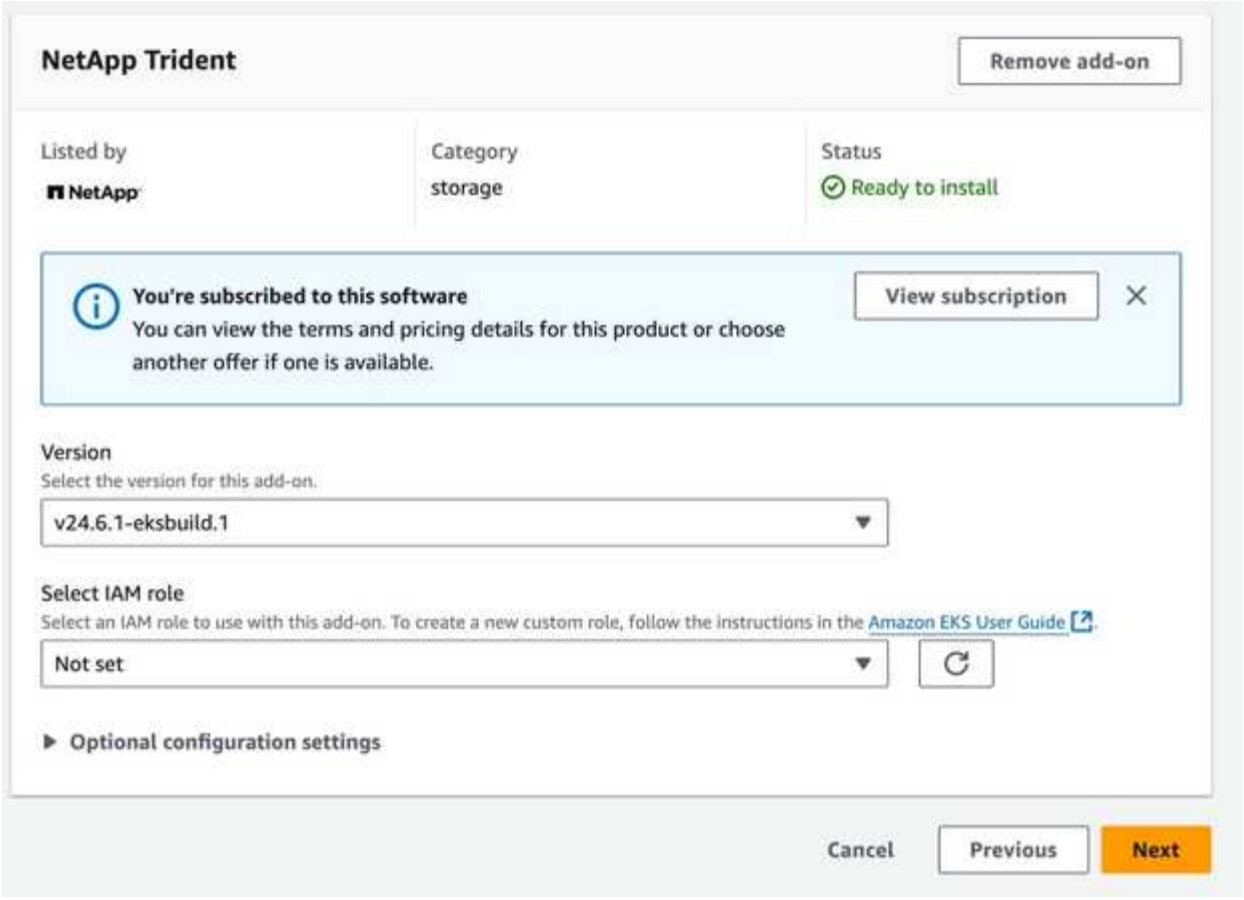
1. Sur votre cluster EKS Kubernetes, accédez à l'onglet **Add-ons**.



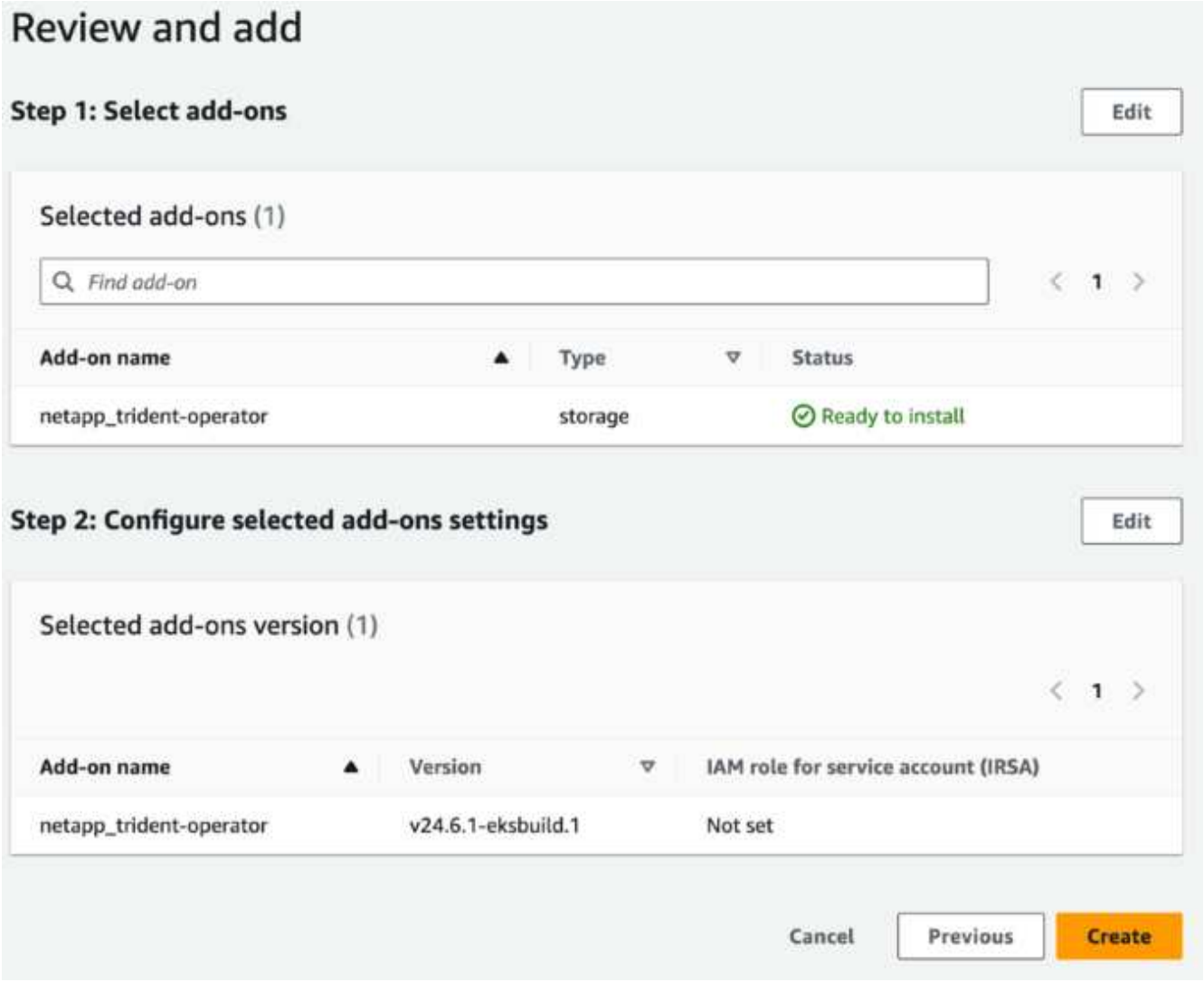
2. Accédez à **add-ons** AWS Marketplace et choisissez la catégorie *Storage*.



3. Localisez **NetApp Trident** et cochez la case du module complémentaire Astra Trident.
4. Choisissez la version souhaitée du module complémentaire.



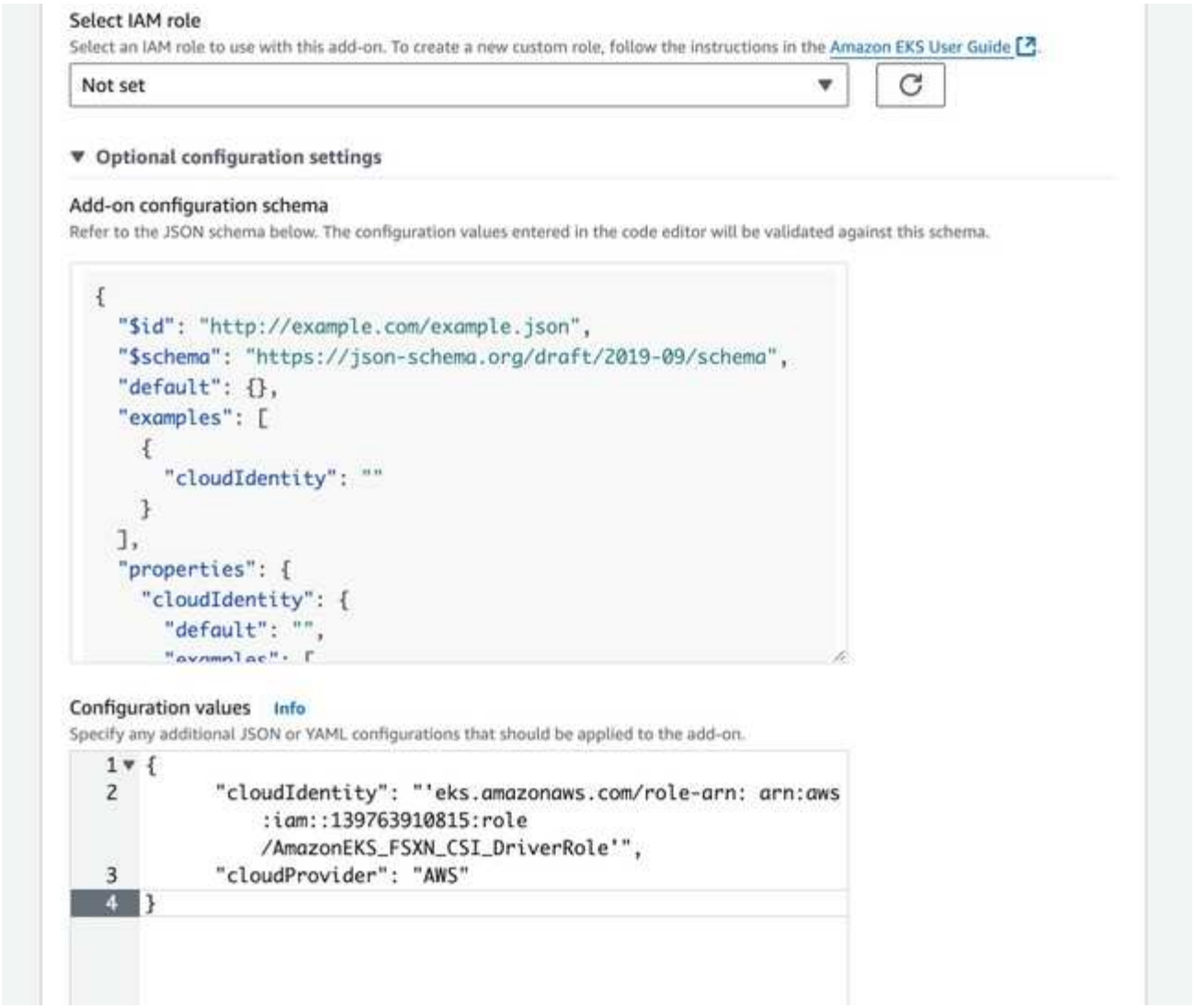
5. Sélectionnez l'option rôle IAM à hériter du nœud.



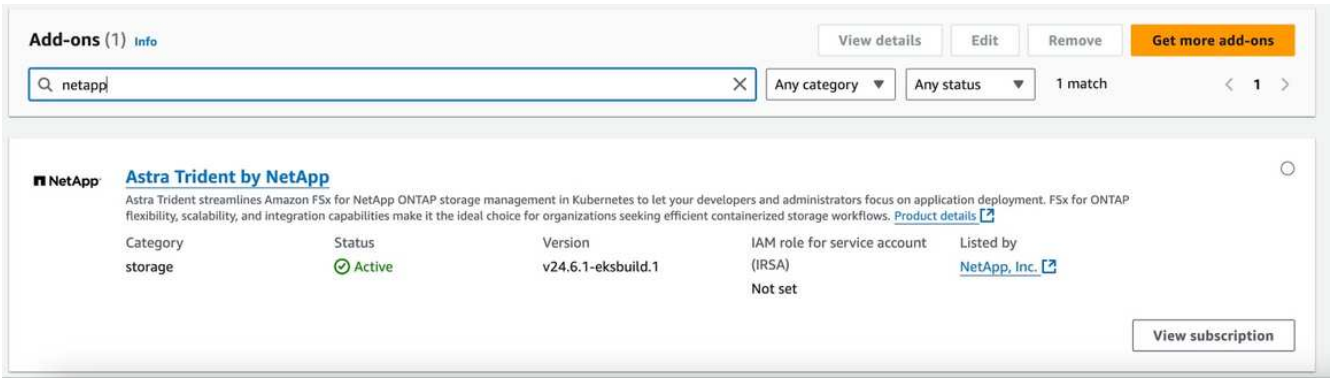
6. (Facultatif) configurez tous les paramètres de configuration facultatifs selon les besoins et sélectionnez **Suivant**.

   Suivez le schéma de configuration **Add-on** et définissez le paramètre configurationValues dans la section **Configuration Values** sur le fil de rôle que vous avez créé à l'étape précédente (la valeur doit être au format suivant : `eks.amazonaws.com/role-arn: arn:aws:iam::464262061435:role/AmazonEKS_FSXN_CSI_DriverRole`). Si vous sélectionnez remplacer pour la méthode de résolution des conflits, un ou plusieurs des paramètres du module complémentaire existant peuvent être remplacés par les paramètres du module complémentaire Amazon EKS. Si vous n'activez pas cette option et qu'il y a un conflit avec vos paramètres existants, l'opération échoue. Vous pouvez utiliser le message d'erreur qui en résulte pour résoudre le conflit. Avant de sélectionner cette option, assurez-vous que le module complémentaire Amazon EKS ne gère pas les paramètres que vous devez gérer vous-même.

   > Lorsque vous configurez le paramètre facultatif `cloudIdentity` , assurez-vous que vous spécifiez `AWS` en tant que `cloudProvider` lors de l'installation de Trident à l'aide du module complémentaire EKS.



7. Sélectionnez **Créer**.
8. Vérifiez que l'état du complément est *Active*.



Installez/désinstallez le module complémentaire Astra Trident EKS à l'aide de l'interface de ligne de commande

Installez le module complémentaire Astra Trident EKS à l'aide de l'interface de ligne de commande :

L'exemple de commande suivant installe le module complémentaire Astra Trident EKS :
`eksctl create addon --cluster K8s-arm --name netapp_trident-operator --version v24.6.1-eksbuild`

`eksctl create addon --cluster clusterName --name netapp_trident-operator --version v24.6.1-eksbuild.1` (avec une version dédiée)

> Lorsque vous configurez le paramètre facultatif `cloudIdentity` , assurez-vous de spécifier `cloudProvider` lors de l'installation de Trident à l'aide du module complémentaire EKS.

Désinstallez le module complémentaire Astra Trident EKS à l'aide de l'interface de ligne de commande :

La commande suivante désinstalle le module complémentaire Astra Trident EKS :
`eksctl delete addon --cluster K8s-arm --name netapp_trident-operator`

Création de systèmes back-end avec kubectl

Un back-end définit la relation entre Trident et un système de stockage. Il explique à Trident comment communiquer avec ce système de stockage et comment Trident doit provisionner les volumes à partir de celui-ci. Une fois Trident installé, l'étape suivante consiste à créer un back-end. La `TridentBackendConfig` définition personnalisée des ressources (CRD) vous permet de créer et de gérer des systèmes back-end Trident directement via l'interface Kubernetes. Pour cela, vous pouvez utiliser `kubectl` ou l'outil CLI équivalent pour votre distribution Kubernetes.

`TridentBackendConfig`

`TridentBackendConfig` (`tbc`, `tbconfig`, `tbackendconfig`) Est un système CRD front-end, qui vous permet de gérer des systèmes Trident back-end à l'aide de `kubectl`. Kubernetes et les administrateurs du stockage peuvent désormais créer et gérer des systèmes back-end directement via l'interface de ligne de commande Kubernetes sans avoir besoin d'un utilitaire de ligne de commande dédié (`tridentctl`).

Lors de la création d'un `TridentBackendConfig` objet :

- Un back-end est créé automatiquement par Trident en fonction de la configuration que vous fournissez. Ceci est représenté en interne sous la forme d'une `TridentBackend` CR(`tbe` , `tridentbackend`).
- Le `TridentBackendConfig` est lié de manière unique à un `TridentBackend` qui a été créé par 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` Les CRS sont créés automatiquement par Trident. Vous ne devez pas les modifier. Si vous souhaitez effectuer des mises à jour vers des systèmes back-end, modifiez l'objet pour procéder `TridentBackendConfig` à cette opération.

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 du pilote de stockage souhaité, reportez-vous au "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`.

Les systèmes back-end créés avec `tridentctl` n'ont pas d'objet associé `TridentBackendConfig`. Vous pouvez choisir de gérer ces systèmes back-end avec `kubectl` en créant une `TridentBackendConfig` demande de modification. Veillez à spécifier des paramètres de configuration identiques (tels que `spec.backendName`, , , `spec.storagePrefix` `spec.storageDriverName` etc.). Trident lie automatiquement le nouveau système créé `TridentBackendConfig` avec le système 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 contient les informations d'identification dont Trident a besoin pour 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: t@Ax@7q(>
```

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

Description des champs secrets de la plate-forme de stockage	Secret	Description des champs
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
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` ID 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, la suppression du entraîne la suppression de Trident, `TridentBackendConfig` ainsi que de la `TridentBackendConfig` demande de modification.
  - 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`: Trident n'est pas en mesure de déterminer l'état ou l'existence du back-end associé à la `TridentBackendConfig` CR. Par exemple, si le serveur d'API ne répond pas ou si le `tridentbackends.trident.netapp.io` CRD est manquant. Cela peut nécessiter une intervention.

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 la `TridentBackendConfig` demande de modification. Le `lastOperationStatus` champ représente l'état de la dernière opération de la `TridentBackendConfig` CR, qui peut être déclenchée par l'utilisateur (par exemple, l'utilisateur a modifié quelque chose dans `spec`) ou déclenchée par Trident (par exemple, lors d'un redémarrage de Trident). Il peut s'agir d'un succès ou d'un échec. `phase` Représente l'état de la relation entre la `TridentBackendConfig` CR et le back-end. Dans l'exemple ci-dessus, `phase` a la valeur liée, ce qui signifie que la `TridentBackendConfig` CR est associée 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".

Gestion des systèmes back-end

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 Trident de supprimer/conserver les systèmes back-end (sur la base de `deletionPolicy` la). Pour supprimer un back-end, assurez-vous que `deletionPolicy` est défini sur supprimer. Pour supprimer uniquement le `TridentBackendConfig`, assurez-vous que `deletionPolicy` est défini sur conserver. Cela permet de s'assurer que le back-end est toujours présent et peut être géré à l'aide de `tridentctl`.

Exécutez la commande suivante :

```
kubectl delete tbc <tbc-name> -n trident
```

Trident ne supprime pas les secrets Kubernetes 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 informations d'identification, le secret Kubernetes utilisé dans l' `TridentBackendConfig`objet doit être mis à jour. Trident met automatiquement à jour le back-end avec les informations d'identification les plus récentes 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.
  - Vous pouvez mettre à jour `TridentBackendConfig` Objets directement dans Kubernetes à l'aide de la commande suivante :

```
kubectl apply -f <updated-backend-file.yaml>
```

- Vous pouvez également apporter des modifications à l'existant `TridentBackendConfig` CR à l'aide de 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 de 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 Trident a provisionné des volumes et des snapshots à partir de ce back-end, la suppression du back-end empêche le provisionnement de nouveaux volumes. 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 méthodes de gestion des systèmes back-end dans Trident.

Options de gestion des systèmes back-end

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 :

- Systèmes back-end existants, sans 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 scénarios, les systèmes back-end continueront d'être présents, avec 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. La clé secrète contient les informations d'identification dont Trident a besoin pour 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 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 [observer l'UUID du back-end].

Étape 1 : confirmer `deletionPolicy` **est défini sur** `retain`

Examinons la valeur de `deletionPolicy`. Ce paramètre doit être défini sur `retain`. Cela garantit que lorsqu'une `TridentBackendConfig` demande de modification est supprimée, la définition du back-end 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 de `TridentBackendConfig` l'objet, Trident le supprime simplement sans réellement supprimer le back-end lui-même.

Créer et gérer des classes de stockage

Créer une classe de stockage

Configurez un objet StorageClass Kubernetes et créez la classe de stockage pour indiquer à Trident comment provisionner les volumes.

Configuration d'un objet StorageClass Kubernetes

Le système "Objet classe de stockage Kubernetes" identifie Trident en tant que mécanisme de provisionnement utilisé pour cette classe et indique à Trident comment provisionner un volume. Par exemple :

```
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
```

Reportez-vous "Kubernetes et objets Trident"à pour plus de détails sur l'interaction des classes de stockage avec les `PersistentVolumeClaim` paramètres et pour le contrôle de la manière dont Trident provisionne les volumes.

Créer une classe de stockage

Après avoir créé l'objet StorageClass, vous pouvez créer la classe de stockage. Échantillons de classe de stockage fournit des exemples de base que vous pouvez utiliser ou modifier.

Étapes

1. Il s'agit d'un objet Kubernetes, alors utilisez-le `kubectl` Pour la créer dans Kubernetes.

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

2. Vous devriez maintenant voir une classe de stockage **Basic-csi** dans Kubernetes et Trident, et Trident aurait dû détecter les pools sur le 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"
        ]
      }
    }
  ]
}
```

Échantillons de classe de stockage

Trident fournit "définition simple de classes de stockage pour des systèmes back-end spécifiques".

Vous pouvez également modifier `sample-input/storage-class-csi.yaml.templ` fichier fourni avec le programme d'installation et de 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
```

Gérer les classes de stockage

Vous pouvez afficher les classes de stockage existantes, définir une classe de stockage par défaut, identifier le back-end de la classe de stockage et supprimer les classes de stockage.

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 de Trident, exécutez la commande suivante :

```
tridentctl get storageclass
```

- Pour afficher le détail de la classe de stockage synchronisée de 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.

> Votre cluster ne doit contenir qu'une seule classe de stockage par défaut à la fois. 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 du type de questions que vous pouvez répondre avec le fichier JSON qui `tridentctl` sort pour les objets backend Trident. Cela utilise l' `jq`utilitaire, que vous devrez peut-être installer en premier.

```
tridentctl get storageclass -o json | jq  '[.items[] | {storageClass:
.Config.name, backends: [.storage]|unique}]'
```

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.

Tous les volumes persistants créés via cette classe de stockage ne sont pas concernés et Trident continuera de les gérer.

> Trident applique un espace vide `fsType` pour les volumes qu'il crée. Pour les systèmes back-end iSCSI, il est recommandé d'appliquer la `parameters.fsType` classe de stockage. Vous devez supprimer les classes de stockage existantes et les recréer avec les classes `parameters.fsType` spécifiées.

Provisionnement et gestion des volumes

Provisionner un volume

Créez un volume persistant et une demande de volume persistant qui utilisent la classe de stockage Kubernetes configurée pour demander l'accès au volume persistant. Vous pouvez ensuite monter le volume persistant sur un pod.

Présentation

A "*Volume persistant*" (PV) est une ressource de stockage physique provisionnée par l'administrateur du cluster sur un cluster Kubernetes. Le "*PersistentVolumeClaim*" (PVC) est une demande d'accès au volume persistant sur le cluster.

Le PVC peut être configuré pour demander un stockage d'une certaine taille ou d'un certain mode d'accès. À l'aide de la classe de stockage associée, l'administrateur du cluster peut contrôler plus que la taille du volume persistant et le mode d'accès, tels que les performances ou le niveau de service.

Après avoir créé le volume persistant et la demande de volume persistant, vous pouvez monter le volume dans un pod.

Exemples de manifestes

Exemple de manifeste de volume persistant

Cet exemple de manifeste montre un volume persistant de base de 10Gi associé à StorageClass `basic-csi`.

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-storage
  labels:
    type: local
spec:
  storageClassName: basic-csi
  capacity:
    storage: 10Gi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: "/my/host/path"
```

Exemples de manifestes de demande de volume persistant

Ces exemples présentent les options de configuration de base de la PVC.

PVC avec accès RWO

Cet exemple montre une demande de volume persistant de base avec accès RWO associée à une classe de stockage nommée `basic-csi`.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc-storage
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: basic-csi
```

PVC avec NVMe/TCP

Cet exemple présente une demande de volume persistant de base pour NVMe/TCP avec accès RWO associée à une classe de stockage nommée `protection-gold`.

```
---
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
name: pvc-san-nvme
spec:
accessModes:
  - ReadWriteOnce
resources:
  requests:
    storage: 300Mi
storageClassName: protection-gold
```

Échantillons de manifeste de pod

Ces exemples présentent les configurations de base pour fixer la demande de volume persistant à un pod.

Configuration de base

```
kind: Pod
apiVersion: v1
metadata:
  name: pv-pod
spec:
  volumes:
    - name: pv-storage
      persistentVolumeClaim:
       claimName: basic
  containers:
    - name: pv-container
      image: nginx
      ports:
        - containerPort: 80
          name: "http-server"
      volumeMounts:
        - mountPath: "/my/mount/path"
          name: pv-storage
```

Configuration NVMe/TCP de base

```
---
apiVersion: v1
kind: Pod
metadata:
  creationTimestamp: null
  labels:
    run: nginx
  name: nginx
spec:
  containers:
    - image: nginx
      name: nginx
      resources: {}
      volumeMounts:
        - mountPath: "/usr/share/nginx/html"
          name: task-pv-storage
  dnsPolicy: ClusterFirst
  restartPolicy: Always
  volumes:
    - name: task-pv-storage
      persistentVolumeClaim:
      claimName: pvc-san-nvme
```

Créer le volume persistant et la demande de volume persistant

Étapes

1. Créer la PV.

```
kubectl create -f pv.yaml
```

2. Vérifiez l'état du PV.

```
kubectl get pv
NAME        CAPACITY  ACCESS MODES  RECLAIM POLICY  STATUS    CLAIM
STORAGECLASS  REASON  AGE
pv-storage  4Gi       RWO           Retain          Available
7s
```

3. Créer la PVC.

```
kubectl create -f pvc.yaml
```

4. Vérifiez l'état de la demande de volume persistant.

```
kubectl get pvc
NAME        STATUS VOLUME     CAPACITY ACCESS MODES STORAGECLASS AGE
pvc-storage Bound  pv-name 2Gi      RWO                       5m
```

5. Montez le volume dans un pod.

```
kubectl create -f pv-pod.yaml
```

> Vous pouvez surveiller la progression à l'aide de `kubectl get pod --watch`.

6. Vérifiez que le volume est monté sur `/my/mount/path`.

```
kubectl exec -it task-pv-pod -- df -h /my/mount/path
```

7. Vous pouvez maintenant supprimer le Pod. L'application Pod n'existera plus, mais le volume restera.

```
kubectl delete pod task-pv-pod
```

Reportez-vous "Kubernetes et objets Trident"à pour plus de détails sur l'interaction des classes de stockage avec les `PersistentVolumeClaim` paramètres et pour le contrôle de la manière dont Trident provisionne les volumes.

Développement des volumes

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.

Étape 1 : configurer la classe de stockage pour prendre en charge l'extension de volume

Modifiez la définition de la classe de stockage pour définir le `allowVolumeExpansion` champ à `true`.

```
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

Modifiez la définition de la demande de volume persistant et mettez à jour le `spec.resources.requests.storage` pour refléter la nouvelle taille souhaitée, qui doit être supérieure à la taille d'origine.

```
cat pvc-ontapsan.yaml
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: san-pvc
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: ontap-san
```

Trident crée un volume persistant et 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

Reliez le volume persistant à un pod pour qu'il soit redimensionné. Lors du redimensionnement d'un volume persistant iSCSI, deux scénarios sont possibles :

- Si le volume persistant est connecté à un pod, Trident étend le volume sur le back-end de stockage, analyse à nouveau le périphérique et redimensionne le système de fichiers.
- Lors d'une tentative de redimensionnement d'un volume persistant non attaché, Trident étend le volume sur le back-end de stockage. 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.

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 fonctionnement correct de l'extension en vérifiant la taille de la demande de volume persistant, du volume PV et du volume 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

Trident prend en charge l'extension de volume des volumes NFS PVS provisionnés sur `ontap-nas`, `ontap-nas-economy`, `ontap-nas-flexgroup` `gcp-cvs` et les `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
```

Trident devrait créer un volume persistant NFS de 20 Mio 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 1 Gio :

```
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 travaillé correctement en vérifiant la taille de la demande de volume persistant, de la valeur PV et du volume 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`.

Présentation et considérations

Vous pouvez importer un volume dans Trident vers :

- Conteneurisation d'une application et réutilisation de son jeu de données existant
- Utilisez un clone d'un jeu de données pour une application éphémère
- Reconstruction d'un cluster Kubernetes en panne
- Migration des données applicatives pendant la reprise après incident

Considérations

Avant d'importer un volume, consultez les considérations suivantes.

- Trident peut importer des volumes ONTAP de type RW (lecture-écriture) uniquement. Les volumes de type DP (protection des données) sont des volumes de destination SnapMirror. Vous devez rompre la relation de miroir avant d'importer le volume dans Trident.

- Nous vous suggérons d'importer des volumes sans connexions actives. Pour importer un volume activement utilisé, clonez-le, puis effectuez l'importation.

C'est particulièrement important pour les volumes en mode bloc, car Kubernetes ignorerait la connexion précédente et pourrait facilement relier un volume actif à un pod. Cela peut entraîner une corruption des données.

- Bien que `StorageClass` doit être spécifié sur une demande de volume persistant, Trident n'utilise pas ce paramètre lors de l'importation. Les classes de stockage sont utilisées lors de la création du volume pour sélectionner un pool disponible en fonction des caractéristiques de stockage. Comme le volume existe déjà, aucune sélection de pool n'est requise pendant l'importation. Par conséquent, l'importation n'échouera pas même si le volume existe sur un back-end ou un pool qui ne correspond pas à la classe de stockage spécifiée dans le PVC.
- La taille du volume existant est déterminée et définie dans la PVC. 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.
- Par défaut, Trident gère la demande de volume persistant et renomme la FlexVol et la LUN sur le back-end. Vous pouvez passer `--no-manage` l'indicateur pour importer un volume non géré. Si vous utilisez `--no-manage`, Trident n'effectue aucune opération supplémentaire sur la PVC ou la PV pour le cycle de vie des objets. Le volume de stockage n'est pas supprimé lorsque le volume persistant est supprimé et d'autres opérations telles que le clone de volume et le redimensionnement de volume 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.

Importer un volume

Vous pouvez utiliser `tridentctl import` pour importer un volume.

Étapes

1. Création du fichier de demande de volume persistant (par exemple, `pvc.yaml`) Qui sera utilisé pour créer la PVC. Le fichier PVC doit inclure `name`, `namespace`, `accessModes`, et `storageClassName`. Vous pouvez également spécifier `unixPermissions` Dans votre définition de PVC.

   Voici un exemple de spécification minimale :

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: my_claim
  namespace: my_namespace
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: my_storage_class
```

N'incluez pas de paramètres supplémentaires tels que le nom du volume persistant ou la taille du volume. Cela peut entraîner l'échec de la commande d'importation.

2. Utilisez `tridentctl import` la commande pour spécifier le nom du back-end Trident contenant le volume et le nom qui identifie de manière unique le volume sur le stockage (par exemple : ONTAP FlexVol, Element Volume, Cloud Volumes Service path). L' `-f`argument est requis pour spécifier le chemin d'accès au fichier PVC.

```
tridentctl import volume <backendName> <volumeName> -f <path-to-pvc-file>
```

Exemples

Consultez les exemples d'importation de volume suivants pour les pilotes pris en charge.

NAS ONTAP et FlexGroup NAS ONTAP

Trident prend en charge l'importation de volumes à l'aide des `ontap-nas` pilotes et `ontap-nas-flexgroup`.

- Le `ontap-nas-economy` 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.

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.

Exemples NAS de ONTAP

Voici un exemple d'importation de volume géré et de volume non géré.

Volume géré

L'exemple suivant importe un volume nommé `managed_volume` sur un système back-end nommé `ontap_nas`:

```
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    |
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
```

Volume non géré

Lors de l'utilisation de l' `--no-manage`argument, Trident ne renomme pas le volume.

L'exemple suivant importe `unmanaged_volume` sur le `ontap_nas` back-end :

```
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   |
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
```

SAN ONTAP

Trident prend en charge l'importation de volumes à l'aide des `ontap-san` pilotes et `ontap-san-economy`.

Trident peut importer des volumes FlexVol ONTAP qui contiennent une seule LUN. Cela est cohérent avec le `ontap-san` pilote, qui crée une FlexVol pour chaque demande de volume persistant et une LUN au sein de la FlexVol. Trident importe le FlexVol et l'associe à la définition du PVC.

Exemples de SAN ONTAP

Voici un exemple d'importation de volume géré et de volume non géré.

Volume géré

Pour les volumes gérés, Trident renomme le FlexVol au `pvc-<uuid>` format et le LUN dans le FlexVol à `lun0`.

L'exemple suivant importe le `ontap-san-managed` FlexVol présent sur le `ontap_san_default` back-end :

```
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    |
+-----------------------------------------+--------+--------------
+----------+-----------------------------------------+--------+---------+
```

Volume non géré

L'exemple suivant importe `unmanaged_example_volume` sur le `ontap_san` back-end :

```
tridentctl import volume -n trident san_blog unmanaged_example_volume
-f pvc-import.yaml --no-manage
+-----------------------------------------+---------+--------------
+----------+-----------------------------------------+--------+---------+
|                   NAME                  |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+-----------------------------------------+---------+--------------
+----------+-----------------------------------------+--------+---------+
| pvc-1fc999c9-ce8c-459c-82e4-ed4380a4b228 | 1.0 GiB | san-blog      |
block    | e3275890-7d80-4af6-90cc-c7a0759f555a | online | false   |
+-----------------------------------------+---------+--------------
+----------+-----------------------------------------+--------+---------+
```

Si des LUN sont mappées à des igroups qui partagent un IQN avec un IQN de nœud Kubernetes, comme dans l'exemple suivant, l'erreur s'affiche : `LUN already mapped to initiator(s) in this group`. Vous devez supprimer l'initiateur ou annuler le mappage de la LUN pour importer le volume.

```
Vserver   Igroup       Protocol OS Type  Initiators
--------- ------------ -------- -------- ------------------------------------
svm0      k8s-nodename.example.com-fe5d36f2-cded-4f38-9eb0-c7719fc2f9f3
                       iscsi    linux    iqn.1994-05.com.redhat:4c2e1cf35e0
svm0      unmanaged-example-igroup
                       mixed    linux    iqn.1994-05.com.redhat:4c2e1cf35e0
```

Elément

Trident prend en charge le logiciel NetApp Element et l'importation de volumes NetApp HCI à l'aide du `solidfire-san` pilote.

> Le pilote d'élément prend en charge les noms de volume dupliqués. Cependant, Trident renvoie une erreur s'il existe des noms de volumes dupliqués. Pour contourner ce problème, clonez le volume, indiquez un nom de volume unique et importez le volume cloné.

Exemple d'élément

L'exemple suivant importe un `element-managed` volume sur le back-end `element_default`.

```
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    |
+----------------------------------------+--------+---------------
+----------+----------------------------------------+--------+---------+
```

Google Cloud Platform

Trident prend en charge l'importation de volumes à l'aide du `gcp-cvs` pilote.

> Pour importer un volume soutenu par NetApp Cloud Volumes Service dans Google Cloud Platform, identifiez le volume par son chemin d'accès au volume. Le chemin du volume est la partie du chemin d'exportation du volume après le `:/`. Par exemple, si le chemin d'exportation est `10.0.0.1:/adroit-jolly-swift`, le chemin du volume est `adroit-jolly-swift`.

Exemple de Google Cloud Platform

L'exemple suivant importe un `gcp-cvs` volume sur le back-end `gcpcvs_YEppr` avec le chemin de volume de `adroit-jolly-swift`.

```
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    |
+-----------------------------------------+--------+--------------
+----------+-----------------------------------------+--------+---------+
```

Azure NetApp Files

Trident prend en charge l'importation de volumes à l'aide du `azure-netapp-files` pilote.

> Pour importer un volume Azure NetApp Files, identifiez-le par son chemin d'accès au volume. Le chemin du volume est la partie du chemin d'exportation du volume après le `:/`. Par exemple, si le chemin de montage est `10.0.0.2:/importvol1`, le chemin du volume est `importvol1`.

Exemple Azure NetApp Files

L'exemple suivant importe un `azure-netapp-files` volume sur le back-end `azurenetappfiles_40517` avec le chemin de volume `importvol1`.

```
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    |
+-----------------------------------------+---------+--------------
+----------+-----------------------------------------+--------+---------+
```

Personnaliser les noms et les étiquettes des volumes

Avec Trident, vous pouvez attribuer des noms et des libellés significatifs aux volumes que vous créez. Vous pouvez ainsi identifier et mapper facilement les volumes vers leurs ressources Kubernetes respectives. Vous pouvez également définir des modèles au niveau du back-end pour créer des noms de volume personnalisés et des étiquettes personnalisées. Tous les volumes que vous créez, importez ou clonez seront conformes aux modèles.

Avant de commencer

Prise en charge des noms et des libellés de volumes personnalisables :

1. Opérations de création, d'importation et de clonage de volumes.
2. Dans le cas du pilote ontap-nas-Economy, seul le nom du volume Qtree est conforme au modèle de nom.
3. Dans le cas d'un pilote ontap-san-Economy, seul le nom de LUN est conforme au modèle de nom.

Limites

1. Les noms de volume personnalisables sont uniquement compatibles avec les pilotes ONTAP sur site.
2. Les noms de volume personnalisables ne s'appliquent pas aux volumes existants.

Comportements clés des noms de volume personnalisables

1. Si une défaillance survient en raison d'une syntaxe incorrecte dans un modèle de nom, la création du back-end échoue. Toutefois, si l'application modèle échoue, le volume sera nommé conformément à la convention de nommage existante.
2. Le préfixe de stockage n'est pas applicable lorsqu'un volume est nommé à l'aide d'un modèle de nom de la configuration back-end. Toute valeur de préfixe souhaitée peut être directement ajoutée au modèle.

Exemples de configuration back-end avec modèle de nom et étiquettes

Les modèles de noms personnalisés peuvent être définis au niveau de la racine et/ou du pool.

Exemple de niveau racine

```
{
"version": 1,
"storageDriverName": "ontap-nas",
"backendName": "ontap-nfs-backend",
"managementLIF": "<ip address>",
"svm": "svm0",
"username": "<admin>",
"password": "<password>",
"defaults": {
    "nameTemplate":
"{{.volume.Name}}_{{.labels.cluster}}_{{.volume.Namespace}}_{{.volume.Requ
estName}}"
},
"labels": {"cluster": "ClusterA", "PVC":
"{{.volume.Namespace}}_{{.volume.RequestName}}"}
}
```

Exemple au niveau du pool

```
{
  "version": 1,
  "storageDriverName": "ontap-nas",
  "backendName": "ontap-nfs-backend",
  "managementLIF": "<ip address>",
  "svm": "svm0",
 "username": "<admin>",
  "password": "<password>",
  "useREST": true,
  "storage": [
  {
      "labels":{"labelname":"label1", "name": "{{ .volume.Name }}"},
      "defaults":
      {
          "nameTemplate": "pool01_{{ .volume.Name }}_{{ .labels.cluster
}}_{{ .volume.Namespace }}_{{ .volume.RequestName }}"
      }
   },
  {
      "labels":{"cluster":"label2", "name": "{{ .volume.Name }}"},
      "defaults":
      {
          "nameTemplate": "pool02_{{ .volume.Name }}_{{ .labels.cluster
}}_{{ .volume.Namespace }}_{{ .volume.RequestName }}"
      }
}
  ]
}
```

Exemples de modèles de noms

Exemple 1 :

```
"nameTemplate": "{{ .config.StoragePrefix }}_{{ .volume.Name }}_{{
.config.BackendName }}"
```

Exemple 2 :

```
"nameTemplate": "pool_{{ .config.StoragePrefix }}_{{ .volume.Name }}_{{
slice .volume.RequestName 1 5 }}""
```

Points à prendre en compte

1. Dans le cas des importations en volume, les étiquettes ne sont mises à jour que si le volume existant comporte des étiquettes dans un format spécifique. Par exemple : `{"provisioning":{"Cluster":"ClusterA", "PVC": "pvcname"}}`.
2. Dans le cas des importations de volume gérées, le nom du volume suit le modèle de nom défini au niveau racine dans la définition du back-end.
3. Trident ne prend pas en charge l'utilisation d'un opérateur de tranche avec le préfixe de stockage.
4. Si les modèles ne donnent pas de noms de volumes uniques, Trident ajoute quelques caractères aléatoires pour créer des noms de volumes uniques.
5. Si le nom personnalisé d'un volume économique NAS dépasse 64 caractères, Trident nomtera les volumes conformément à la convention de nommage existante. Pour tous les autres pilotes ONTAP, si le nom du volume dépasse la limite de nom, le processus de création du volume échoue.

Partager un volume NFS entre les espaces de noms

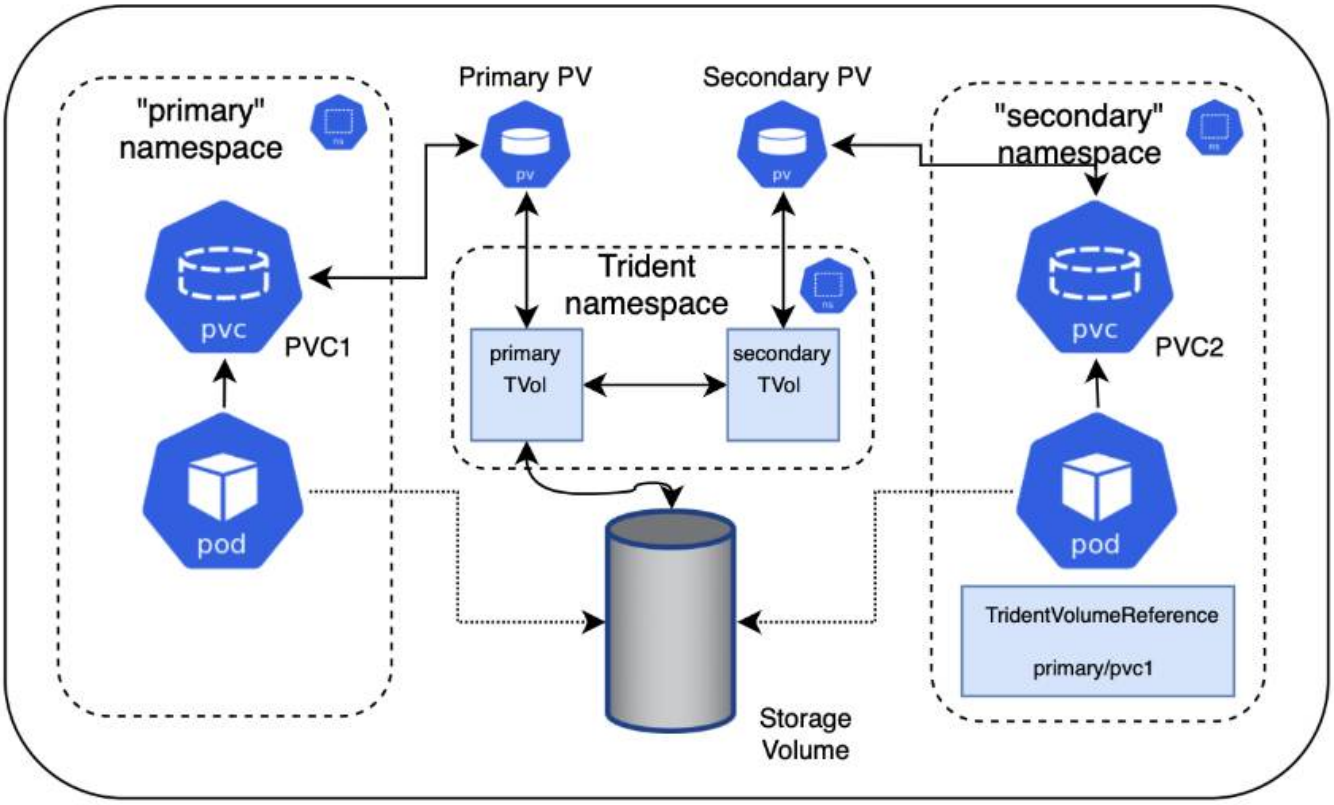
Avec Trident, vous pouvez créer un volume dans un espace de noms principal et le partager dans un ou plusieurs espaces de noms secondaires.

Caractéristiques

Le système TridentVolumeReference CR vous permet de partager en toute sécurité des volumes ReadWriteMaly (RWX) NFS 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
   ```

   Trident crée le volume de stockage PV 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

Trident lit l' `shareFromPVC`annotation sur la demande de volume de destination et crée le volume persistant de destination en tant que volume subordonné sans ressource de stockage propre qui pointe vers le volume persistant source et partage la ressource de stockage du volume persistant 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. Trident supprime l'accès au volume sur l'espace de noms source et maintient l'accès aux autres espaces de noms qui partagent le volume. Lorsque tous les espaces de noms qui référencent le volume sont supprimés, 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

- 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".

Utiliser la topologie CSI

Trident peut créer et attacher de manière sélective des volumes aux nœuds présents dans un cluster Kubernetes en utilisant le "Fonction de topologie CSI".

Présentation

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. Pour faciliter le provisionnement des volumes pour les charges de travail dans une architecture multi-zone, Trident utilise la topologie CSI.

En savoir plus sur la fonction de topologie CSI "ici".

Kubernetes propose deux modes de liaison de volumes :

- Avec la `VolumeBindingMode` valeur définie sur `Immediate`, Trident crée le volume sans connaissance 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 de contraintes de topologie. Les volumes persistants sont créés sans dépendance vis-à-vis des exigences de planification du pod demandeur.

- 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 de prendre en compte la topologie (`topology.kubernetes.io/region`et `topology.kubernetes.io/zone`). Ces libellés **doivent être présents sur les nœuds du cluster** avant l'installation de Trident pour que 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 Trident peuvent être conçus pour provisionner de manière sélective des volumes en fonction des zones de disponibilité. Chaque back-end peut porter un bloc facultatif `supportedTopologies` représentant une liste de zones et de régions 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` est utilisé pour fournir une liste de régions et de zones par back-end. 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 des régions et zones fournies dans un back-end, Trident crée un volume sur le back-end.

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
  supportedTopologies:
  - topology.kubernetes.io/region: us-central1
    topology.kubernetes.io/zone: us-central1-a
- labels:
    workload: dev
  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 fournie ci-dessus, `volumeBindingMode` est définie 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. La `netapp-san-us-east1` classe de stockage crée des ESV sur le `san-backend-us-east1` 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

Les copies Snapshot de volume Kubernetes de volumes persistants (PVS) permettent d'effectuer des copies instantanées de volumes. Vous pouvez créer un snapshot d'un volume créé à l'aide de Trident, importer un snapshot créé en dehors de Trident, créer un volume à partir d'un snapshot existant et restaurer des données de volume à partir de snapshots.

Présentation

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) pour pouvoir utiliser les snapshots. 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 Déployer 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.

Créer un snapshot de volume

Étapes

1. Créer un `VolumeSnapshotClass`. Pour plus d'informations, reportez-vous à la section "VolumeSnapshotClass".
   - Le `driver` signale au conducteur Trident CSI.
   - `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.

     Exemple

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

2. Créer un snapshot d'une demande de volume persistant existante.

   Exemples

   - 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
     ```

   - Cet exemple crée un objet de snapshot de volume pour une demande de volume persistant nommée `pvc1` et le nom du snapshot est défini sur `pvc1-snap`. Un instantané VolumeSnapshot est similaire à une demande de volume persistant et est associé à une `VolumeSnapshotContent` objet qui représente le snapshot réel.

```
kubectl create -f snap.yaml
volumesnapshot.snapshot.storage.k8s.io/pvc1-snap created

kubectl get volumesnapshots
NAME                    AGE
pvc1-snap               50s
```

- Vous pouvez identifier le `VolumeSnapshotContent` objet pour le `pvc1-snap` VolumeSnapshot en le décrivant. 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.

```
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
.
.
```

Créer une demande de volume persistant à partir d'un snapshot de volume

Vous pouvez utiliser `dataSource` Pour créer une demande de volume persistant à l'aide d'un VolumeSnapshot nommé `<pvc-name>` comme source des données. Une fois la demande de volume persistant créée, elle peut être connectée à un pod et utilisée comme n'importe quel autre PVC.

> La demande de volume sera créée dans le même back-end que le volume source. Reportez-vous à la section "Base de connaissances : la création d'une demande de volume persistant à partir d'un Snapshot de volume persistant Trident ne peut pas être créée dans un autre back-end".

L'exemple suivant crée la demande de volume persistant à l'aide de `pvc1-snap` comme source de données.

```
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
```

Importer un instantané de volume

Trident prend en charge la "Processus Snapshot préprovisionné Kubernetes"permettant à l'administrateur du cluster de créer un `VolumeSnapshotContent` objet et d'importer des snapshots créés en dehors de Trident.

Avant de commencer

Trident doit avoir créé ou importé le volume parent du snapshot.

Étapes

1. **Cluster admin:** Créez un `VolumeSnapshotContent` objet qui fait référence au snapshot back-end. Ceci lance le flux de travail de snapshot dans Trident.
   - Spécifiez le nom du snapshot back-end dans `annotations` comme `trident.netapp.io/internalSnapshotName: <"backend-snapshot-name">`.
   - Spécifiez `<name-of-parent-volume-in-trident>/<volume-snapshot-content-name>` dans `snapshotHandle`. il s'agit de la seule information fournie à Trident par le snapshotter externe dans l' `ListSnapshots`appel.

     Le `<volumeSnapshotContentName>` Impossible de toujours faire correspondre le nom du snapshot back-end en raison des contraintes de dénomination CR.

     Exemple

     L'exemple suivant crée un `VolumeSnapshotContent` objet qui fait référence au snapshot back-end `snap-01`.

```
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshotContent
metadata:
  name: import-snap-content
  annotations:
    trident.netapp.io/internalSnapshotName: "snap-01"  # This is the
name of the snapshot on the backend
spec:
  deletionPolicy: Retain
  driver: csi.trident.netapp.io
  source:
    snapshotHandle: pvc-f71223b5-23b9-4235-bbfe-e269ac7b84b0/import-
snap-content # <import PV name or source PV name>/<volume-snapshot-
content-name>
```

2. **Cluster admin:** Créez le `VolumeSnapshot` CR qui fait référence au `VolumeSnapshotContent` objet. Cette opération demande l'accès à `VolumeSnapshot` dans un espace de noms donné.

   Exemple

   L'exemple suivant crée un `VolumeSnapshot` CR nommée `import-snap` qui fait référence au `VolumeSnapshotContent` nommé `import-snap-content`.

   ```
   apiVersion: snapshot.storage.k8s.io/v1
   kind: VolumeSnapshot
   metadata:
     name: import-snap
   spec:
     # volumeSnapshotClassName: csi-snapclass (not required for pre-
   provisioned or imported snapshots)
     source:
       volumeSnapshotContentName: import-snap-content
   ```

3. **Traitement interne (aucune action requise):** le snapshotter externe reconnaît le nouveau créé `VolumeSnapshotContent` et exécute l' `ListSnapshots`appel. Trident crée le `TridentSnapshot`.

   - Le snapshotter externe définit le `VolumeSnapshotContent` à `readyToUse` et le `VolumeSnapshot` à `true`.
   - Retour Trident `readyToUse=true`.

4. **Tout utilisateur :** Créer un `PersistentVolumeClaim` pour référencer le nouveau `VolumeSnapshot`, où `spec.dataSource` (ou `spec.dataSourceRef`) nom est le `VolumeSnapshot` nom.

   Exemple

   L'exemple suivant crée un PVC faisant référence à `VolumeSnapshot` nommé `import-snap`.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-from-snap
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: simple-sc
  resources:
    requests:
      storage: 1Gi
  dataSource:
    name: import-snap
    kind: VolumeSnapshot
    apiGroup: snapshot.storage.k8s.io
```

Restaurez les données de volume à l'aide de snapshots

Le répertoire des snapshots est masqué par défaut pour faciliter la compatibilité maximale des volumes provisionnés à l'aide de `ontap-nas` et `ontap-nas-economy` pilotes. Activez le `.snapshot` répertoire permettant de restaurer directement les données à partir de snapshots.

Utilisez l'interface de ligne de commandes ONTAP de restauration de snapshot de volume pour restaurer un volume à un état enregistré dans un snapshot précédent.

```
cluster1::*> volume snapshot restore -vserver vs0 -volume vol3 -snapshot
vol3_snap_archive
```

Lorsque vous restaurez une copie Snapshot, la configuration de volume existante est écrasée. Les modifications apportées aux données de volume après la création de la copie Snapshot sont perdues.

Supprimez un volume persistant avec les snapshots associés

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 ». Supprimez les snapshots de volume pour supprimer le volume Trident.

Déployer 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 de snapshot.

```
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
```

Si nécessaire, ouvrir `deploy/kubernetes/snapshot-controller/rbac-snapshot-controller.yaml` et mettre à jour `namespace` à votre espace de noms.

Liens connexes

- "Snapshots de volume"
- "VolumeSnapshotClass"

Gérer et surveiller Trident

Mettez à niveau Trident

Mettez à niveau Trident

À compter de la version 24.02, Trident suit une cadence de quatre mois pour publier trois versions majeures chaque année civile. Chaque nouvelle version exploite les versions précédentes et fournit de nouvelles fonctionnalités, des améliorations de performances, 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 de Trident.

Considérations avant la mise à niveau

Lorsque vous effectuez une mise à niveau vers la dernière version de Trident, tenez compte des points suivants :

- Il ne doit y avoir qu'une seule instance Trident installée sur tous les namespaces d'un cluster Kubernetes donné.
- Trident 23.07 et versions ultérieures requièrent des instantanés de volume v1 et ne prend plus en charge les instantanés alpha ou bêta.
- Si vous avez créé Cloud Volumes Service pour Google Cloud dans le "Type de service CVS", vous devez mettre à jour la configuration back-end pour utiliser le `standardsw` niveau de service ou `zoneredundantstandardsw` lors de la mise à niveau à partir de Trident 23.01. L'échec de la mise à jour du système `serviceLevel` dans le back-end peut entraîner l'échec des volumes. Voir "Exemples de type de service CVS" pour plus de détails.
- Lors de la mise à niveau, il est important que vous fournissez `parameter.fsType` dans `StorageClasses` utilisé par Trident. Vous pouvez supprimer et recréer des données `StorageClasses` sans interrompre les volumes préexistants.
  - Il s'agit d'une exigence ** pour l'application "contextes de sécurité" Pour les volumes SAN.
  - Le répertoire d'entrée sample contient des exemples, tels que `storage-class-basic.yaml.templ` et `storage-class-bronze-default.yaml`.
  - Pour plus d'informations, reportez-vous à la section "Problèmes connus".

Étape 1 : sélectionnez une version

Les versions Trident suivent une convention de dénomination basée sur la date `YY.MM`, où « YY » correspond aux deux derniers chiffres de l'année et « MM » au mois. Les versions de points suivent une `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 une mise à niveau directe de la version 23.04 (ou de toute version 23.04 points) vers la version 24.06.
- Si vous effectuez une mise à niveau à partir d'une version en dehors de la fenêtre à quatre versions, effectuez une mise à niveau en plusieurs étapes. Suivez les instructions de mise à "version antérieure" niveau de pour effectuer la mise à niveau vers la version la plus récente qui s'adapte à la fenêtre à quatre versions. Par exemple, si vous exécutez 22.01 et que vous souhaitez effectuer une mise à niveau vers

24.06 :

a. Première mise à niveau de 22.07 à 23.04.

b. Puis passez de 23.04 à 24.06.

> Lorsque vous effectuez une mise à niveau avec l'opérateur Trident sur OpenShift Container Platform, vous devez effectuer une mise à niveau vers Trident 21.01.1 ou une 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, reportez-vous au "Consultez le document GitHub pour plus d'informations".

Étape 2 : déterminer la méthode d'installation d'origine

Pour déterminer la version que vous avez utilisée pour installer Trident à l'origine :

1. Utiliser `kubectl get pods -n trident` pour examiner les pods.
   - S'il n'y a pas de module opérateur, Trident a été installé à l'aide de `tridentctl`.
   - S'il existe un module opérateur, Trident a été installé à l'aide de l'opérateur Trident soit manuellement, soit à l'aide de l'assistant.
2. S'il y a un module opérateur, utilisez `kubectl describe torc` pour déterminer si Trident a été installé à l'aide de l'assistant.
   - S'il y a une étiquette Helm, Trident a été installé à l'aide de Helm.
   - S'il n'y a pas d'étiquette Helm, Trident a été installé manuellement à l'aide de l'opérateur Trident.

Étape 3 : sélectionnez une méthode de mise à niveau

En général, vous devez "passer d'une méthode d'installation à l'autre"effectuer une mise à niveau en utilisant la même méthode que celle utilisée pour l'installation initiale, mais vous pouvez . Il existe deux options pour mettre à niveau Trident.

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

> Nous vous suggérons de revoir "Comprendre le workflow de mise à niveau de l'opérateur" avant la mise à niveau avec l'opérateur.

*

Mise à niveau avec l'opérateur

Comprendre le workflow de mise à niveau de l'opérateur

Avant d'utiliser l'opérateur Trident pour mettre à niveau Trident, vous devez comprendre les processus en arrière-plan qui se produisent pendant la mise à niveau. Cela inclut les modifications apportées au contrôleur Trident, au pod du contrôleur et aux pods des nœuds, ainsi qu'au jeu de démonstration des nœuds qui activent les mises à jour en continu.

Gestion des mises à niveau par l'opérateur Trident

L'une des nombreuses "Avantages de l'utilisation de l'opérateur Trident" à installer et à mettre à niveau Trident

est la gestion automatique des objets Trident et Kubernetes sans interrompre les volumes montés existants. De cette façon, Trident peut prendre en charge les mises à niveau sans temps d'indisponibilité, ou "*mises à jour en continu*". En particulier, l'opérateur Trident communique avec le cluster Kubernetes pour :

- Supprimez et recréez le déploiement du contrôleur Trident et le nœud DemonSet.
- Remplacez l'afficheur de contrôleur Trident et les pods de nœud Trident par de nouvelles versions.
  - Si un nœud n'est pas mis à jour, il n'empêche pas la mise à jour des nœuds restants.
  - Seuls les nœuds exécutant Trident Node Pod peuvent monter des volumes.

Pour plus d'informations sur l'architecture Trident sur le cluster Kubernetes, reportez-vous à "Architecture Trident"la .

Workflow de mise à niveau de l'opérateur

Lorsque vous lancez une mise à niveau avec l'opérateur Trident :

1. L'opérateur **Trident** :
   a. Détecte la version actuellement installée de Trident (version *n*).
   b. Mise à jour de tous les objets Kubernetes, y compris les CRD, RBAC et le service Trident.
   c. Supprime le déploiement du contrôleur Trident pour la version *n*.
   d. Crée le déploiement du contrôleur Trident pour la version *n+1*.
2. **Kubernetes** crée le pod du contrôleur Trident pour *n+1*.
3. L'opérateur **Trident** :
   a. Supprime le jeu de démonstration du nœud Trident pour *n*. L'opérateur n'attend pas la fin de Node Pod.
   b. Crée le dédéclenchement du nœud Trident pour *n+1*.
4. **Kubernetes** crée des pods de nœuds Trident sur les nœuds qui n'exécutent pas Trident Node Pod *n*. Cela permet de garantir qu'il n'y a jamais plus d'un pod de nœuds Trident, quelle que soit la version, sur un nœud.

Mettez à niveau une installation Trident à l'aide de l'opérateur Trident ou de Helm

Vous pouvez mettre à niveau Trident à l'aide de l'opérateur Trident manuellement ou à l'aide d'Helm. Vous pouvez effectuer une mise à niveau d'une installation opérateur Trident vers une autre installation opérateur Trident ou passer d'une `tridentctl` installation à une version opérateur Trident. Révision "Sélectionnez une méthode de mise à niveau" avant la mise à niveau d'une installation de l'opérateur Trident.

Mettre à niveau une installation manuelle

Vous pouvez effectuer une mise à niveau d'une installation d'opérateur Trident dont le périmètre est défini dans le cluster vers une autre installation d'opérateur Trident dont le périmètre est défini dans le cluster. Toutes les versions de Trident 21.01 et supérieures utilisent un opérateur cluster-scoped.

Pour mettre à niveau à partir de Trident qui a été installé à l'aide de l'opérateur Namespace-scoped (versions 20.07 à 20.10), utilisez les instructions de mise à niveau pour "votre version installée" de Trident.

Description de la tâche

Trident fournit un fichier bundle que vous pouvez utiliser pour installer l'opérateur et créer les objets associés pour votre version Kubernetes.

- Pour les clusters exécutant Kubernetes 1.24, utilisez "bundle_pre_1_25.yaml".
- Pour les clusters exécutant Kubernetes 1.25 ou version ultérieure, utilisez "bundle_post_1_25.yaml".

Avant de commencer

Assurez-vous d'utiliser un cluster Kubernetes en cours d'exécution "Version Kubernetes prise en charge".

Étapes

1. Vérifiez votre version de Trident :

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

2. Supprimez l'opérateur Trident qui a été utilisé pour installer l'instance Trident actuelle. Par exemple, si vous mettez à niveau depuis 23.07, exécutez la commande suivante :

```
kubectl delete -f 23.07.0/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 Trident à l'aide du fichier YAML de bundle approprié pour votre environnement, où *<bundle.yaml>* est `bundle_pre_1_25.yaml` ou `bundle_post_1_25.yaml` basé sur votre version de Kubernetes. Par exemple, si vous installez Trident 24.10, exécutez la commande suivante :

```
kubectl create -f 24.10.0/trident-installer/deploy/<bundle.yaml> -n
trident
```

Mettre à niveau une installation Helm

Vous pouvez mettre à niveau une installation Trident Helm.

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

Si vous avez déjà mis à niveau votre cluster Kubernetes de 1.24 à 1.25 sans mettre à niveau le contrôleur Trident Helm, la mise à niveau Helm échoue. Pour effectuer la mise à niveau de Helm, effectuez les étapes suivantes en tant que conditions préalables :

1. Installez le plug-in Helm-mapkubeapis à partir de https://github.com/helm/helm-mapkubeapis.
2. Effectuez une exécution à sec pour la version Trident dans l'espace de nom où Trident est installé. Cette liste répertorie les ressources qui seront nettoyées.

```
helm mapkubeapis --dry-run trident  --namespace trident
```

3. Effectuez une analyse complète avec Helm pour effectuer le nettoyage.

```
helm mapkubeapis trident  --namespace trident
```

Étapes

1. Si vous "Installez Trident à l'aide de Helm - effectué", vous pouvez utiliser `helm upgrade trident netapp-trident/trident-operator --version 100.2410.0` pour effectuer une mise à niveau en une seule étape. Si vous n'avez pas ajouté le Helm repo ou si vous ne pouvez pas l'utiliser pour mettre à niveau :
   a. Téléchargez la dernière version de Trident sur "La section *Assets* sur GitHub".
   b. Utilisez `helm upgrade` la commande où reflète la version vers laquelle `trident-operator-24.10.0.tgz` vous souhaitez effectuer la mise à niveau.

```
helm upgrade <name> trident-operator-24.10.0.tgz
```

> Si vous définissez des options personnalisées lors de l'installation initiale (par exemple, si vous spécifiez des registres privés en miroir pour les images Trident et CSI), ajoutez le `helm upgrade` commande à l'aide de `--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.

2. 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.

Mise à niveau à partir d'un `tridentctl` Installation sur l'opérateur Trident

Vous pouvez effectuer la mise à niveau vers la dernière version de l'opérateur Trident à partir d'un `tridentctl` installation. Les systèmes back-end et ESV existants seront automatiquement disponibles.

> Avant de passer d'une méthode d'installation à l'autre, consultez "Passage d'une méthode d'installation à l'autre".

Étapes

1. Téléchargez la dernière version de Trident.

```
# Download the release required [24.10.0]
mkdir 24.10.0
cd 24.10.0
wget
https://github.com/NetApp/trident/releases/download/v24.10.0/trident-
installer-24.10.0.tar.gz
tar -xf trident-installer-24.10.0.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 cluster-scoped dans le même namespace.

```
kubectl create -f deploy/<bundle-name.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. Créez une `TridentOrchestrator` CR pour installer Trident.

```
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
```

5. Vérifiez que Trident a été mis à niveau vers la version prévue.

```
kubectl describe torc trident | grep Message -A 3

Message:                Trident installed
Namespace:              trident
Status:                 Installed
Version:                v24.10.0
```

Mise à niveau avec tridentctl

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

Description de la tâche

La désinstallation et la réinstallation de Trident agit comme une mise à niveau. Lorsque vous désinstallez Trident, la demande de volume persistant et le volume persistant utilisés par le déploiement Trident ne sont pas supprimés. Les volumes persistants déjà provisionnés restent disponibles pendant que Trident est hors ligne et Trident provisionne les volumes pour toutes les demandes de volume persistant créées entre la remise en ligne.

Avant de commencer

Révision "Sélectionnez une méthode de mise à niveau" avant la mise à niveau avec `tridentctl`.

Étapes

1. Exécutez la commande de désinstallation dans `tridentctl` pour supprimer toutes les ressources associées à Trident, à l'exception des CRD et des objets associés.

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

2. Réinstallez Trident. Reportez-vous à la "Installez Trident à l'aide de tridentctl".

> N'interrompez pas le processus de mise à niveau. Assurez-vous que le programme d'installation s'exécute jusqu'à la fin.

Gérez Trident à l'aide de tridentctl

Le "Pack d'installation Trident" comprend l' `tridentctl`utilitaire de ligne de commande qui permet un accès simple à Trident. Les utilisateurs Kubernetes disposant de suffisamment de Privileges peuvent l'utiliser pour installer Trident ou gérer le namespace qui contient le pod Trident.

Commandes et indicateurs globaux

Vous pouvez exécuter `tridentctl help` pour obtenir une liste des commandes disponibles pour `tridentctl` ou ajoutez le `--help` marquer n'importe quelle commande pour obtenir une liste d'options et d'indicateurs pour cette commande spécifique.

```
tridentctl [command] [--optional-flag]
```

L'utilitaire Trident `tridentctl` prend en charge les commandes et indicateurs globaux suivants.

Commandes

`create`
Ajouter une ressource à Trident.

`delete`
Supprimez une ou plusieurs ressources de Trident.

`get`
Obtenez une ou plusieurs ressources de Trident.

`help`
Aide sur n'importe quelle commande.

`images`
Imprimez un tableau des images de conteneur dont Trident a besoin.

`import`
Importer une ressource existante dans Trident.

`install`
Installation de Trident.

`logs`
Imprimez les journaux depuis Trident.

`send`
Envoyer une ressource à partir de Trident.

`uninstall`
Désinstallez Trident.

`update`
Modifier une ressource dans Trident.

`update backend state`
Suspendre temporairement les opérations back-end.

`upgrade`
Mettre à niveau une ressource dans Trident.

`version`
Imprimez la version de Trident.

Alarmes globales

-d, --debug

Sortie de débogage.

-h, --help

Aide pour `tridentctl`.

-k, --kubeconfig string

Spécifiez le `KUBECONFIG` Chemin d'accès pour exécuter des commandes localement ou d'un cluster Kubernetes vers un autre.

Vous pouvez également exporter le `KUBECONFIG` Variable permettant de pointer vers un cluster Kubernetes spécifique et de résoudre un problème `tridentctl` commandes pour ce cluster.

-n, --namespace string

Espace de noms du déploiement 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 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.

Options et indicateurs de commande

création

Utilisez `create` la commande pour ajouter une ressource à Trident.

```
tridentctl create [option]
```

Options

`backend`: Ajouter un backend à Trident.

supprimer

Utilisez `delete` la commande pour supprimer une ou plusieurs ressources de Trident.

```
tridentctl delete [option]
```

Options

`backend`: Supprimez un ou plusieurs systèmes back-end de Trident.
`snapshot`: Supprimer un ou plusieurs instantanés de volume de Trident.

`storageclass`: Supprimer une ou plusieurs classes de stockage de Trident.
`volume`: Supprimer un ou plusieurs volumes de stockage de Trident.

obtenez

Utilisez `get` la commande pour obtenir une ou plusieurs ressources de Trident.

```
tridentctl get [option]
```

Options

`backend`: Obtenez un ou plusieurs systèmes back-end de stockage Trident.
`snapshot`: Obtenir un ou plusieurs instantanés de Trident.
`storageclass`: Obtenir une ou plusieurs classes de stockage de Trident.
`volume`: Obtenir un ou plusieurs volumes de Trident.

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.

images

Utilisez `images` des indicateurs pour imprimer un tableau des images de conteneur dont Trident a besoin.

```
tridentctl images [flags]
```

Alarmes

`-h`, `--help`: Aide pour les images.
`-v`, `--k8s-version string`: Version sémantique du cluster Kubernetes.

importer le volume

Utiliser `import volume` la commande pour importer un volume existant dans Trident.

```
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.

installer

Utilisez les `install` indicateurs pour installer Trident.

```
tridentctl install [flags]
```

Alarmes

`--autosupport-image string`: L'image conteneur pour la télémétrie AutoSupport (par défaut "NetApp/Trident AutoSupport:<current-version>").
`--autosupport-proxy string`: Adresse/port d'un proxy pour l'envoi de la télémétrie AutoSupport.
`--enable-node-prep`: Tentative d'installation des modules 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 d'expiration de la requête HTTP pour l'API REST du contrôleur Trident (1m30s par défaut).
`--image-registry string`: Adresse/port d'un registre d'images interne.
`--k8s-timeout duration`: Délai d'expiration pour toutes les opérations Kubernetes (3m0s par défaut).
`--kubelet-dir string`: Emplacement de l'hôte de l'état interne de kubelet (par défaut "/var/lib/kubelet").
`--log-format string`: Le format d'enregistrement Trident (texte, json) (par défaut "texte").
`--node-prep`: Permet à Trident de préparer les nœuds du cluster Kubernetes pour la gestion des volumes à l'aide du protocole de stockage de données spécifié. **Actuellement, `iscsi` est la seule valeur prise en charge.**
`--pv string`: le nom du PV hérité utilisé par Trident, s'assure que cela n'existe pas (par défaut "Trident").
`--pvc string`: Le nom du PVC existant utilisé par Trident, s'assure qu'il n'existe pas (par défaut "Trident").
`--silence-autosupport`: N'envoyez pas automatiquement de paquets AutoSupport à NetApp (valeur par défaut true).
`--silent`: Désactivez la plupart des sorties pendant l'installation.
`--trident-image string`: L'image Trident à installer.
`--use-custom-yaml`: Utilisez tous les fichiers YAML existants qui existent dans le répertoire d'installation.
`--use-ipv6`: Utiliser IPv6 pour la communication de Trident.

journaux

Utilisez `logs` des indicateurs pour imprimer les journaux à partir de Trident.

```
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`: Journal Trident à afficher. L'une des options Trident|auto|Trident-operator|All (par défaut, « auto »).
`--node string`: Nom du nœud Kubernetes à partir duquel collecter les journaux du pod du nœud.
`-p, --previous`: Si elle existe, obtenez les journaux de l'instance de conteneur précédente.
`--sidecars`: Obtenir les billes pour les conteneurs sidecar.

envoyer

Utilisez `send` la commande pour envoyer une ressource à partir de Trident.

```
tridentctl send [option]
```

Options

`autosupport`: Envoyez une archive AutoSupport à NetApp.

désinstaller

Utilisez `uninstall` des indicateurs pour désinstaller Trident.

```
tridentctl uninstall [flags]
```

Alarmes

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

mise à jour

Utiliser `update` la commande pour modifier une ressource dans Trident.

```
tridentctl update [option]
```

Options

`backend`: Mettre à jour un backend dans Trident.

mettre à jour l'état back-end

Utilisez le `update backend state` pour suspendre ou reprendre les opérations back-end.

```
tridentctl update backend state <backend-name> [flag]
```

Points à prendre en compte

- Si un backend est créé à l'aide d'une TridentBackendConfig (tbc), le backend ne peut pas être mis à jour à l'aide d'un `backend.json` fichier.
- Si le `userState` a été défini dans un tbc, il ne peut pas être modifié à l'aide de la `tridentctl update backend state <backend-name> --user-state suspended/normal` commande.
- Pour rétablir la possibilité de définir le `userState` via tridentctl après avoir été défini via tbc, le `userState` champ doit être supprimé du tbc. Cela peut être fait à l'aide de la `kubectl edit tbc` commande. Une fois le `userState` champ supprimé, vous pouvez utiliser `tridentctl update backend state` la commande pour modifier le `userState` d'un back-end.
- Utilisez les `tridentctl update backend state` pour modifier le `userState`. Vous pouvez également mettre à jour le `userState` fichier en utilisant `TridentBackendConfig` ou `backend.json` ; ceci déclenche une réinitialisation complète du back-end et peut prendre du temps.

  #### Alarmes

  `-h, --help`: Aide pour l'état back-end.
  `--user-state`: Défini sur `suspended` pour interrompre les opérations back-end. Réglez sur `normal` pour reprendre les opérations back-end. Lorsqu'il est réglé sur `suspended`:

- `AddVolume` et `Import Volume` sont en pause.
- `CloneVolume`, `ResizeVolume` `PublishVolume`, , `UnPublishVolume`, `CreateSnapshot` `GetSnapshot` `RestoreSnapshot`, `DeleteSnapshot`, , `RemoveVolume`, `GetVolumeExternal` `ReconcileNodeAccess` et restent disponibles.

Vous pouvez également mettre à jour l'état du back-end à l'aide du `userState` champ dans le fichier de configuration du back-end `TridentBackendConfig` ou `backend.json`. Pour plus d'informations, reportez-

vous à "Options de gestion des systèmes back-end" et "Effectuer la gestion back-end avec kubectl".

Exemple:

JSON

Procédez comme suit pour mettre à jour `userState` à l'aide du `backend.json` fichier :

1. Modifiez le `backend.json` fichier pour inclure le `userState` champ avec sa valeur définie sur « terminé ».
2. Mettez à jour le backend à l'aide de la `tridentctl backend update` commande et du chemin d'accès au fichier mis à jour `backend.json` .

   Exemple : `tridentctl backend update -f /<path to backend JSON file>/backend.json`

```
{
    "version": 1,
    "storageDriverName": "ontap-nas",
    "managementLIF": "<redacted>",
    "svm": "nas-svm",
    "backendName": "customBackend",
    "username": "<redacted>",
    "password": "<redacted>",
    "userState": "suspended",
}
```

YAML

Vous pouvez modifier la commande tbc une fois qu'elle a été appliquée à l'aide de la `kubectl edit <tbc-name> -n <namespace>` commande. L'exemple suivant met à jour l'état back-end pour qu'il soit suspendu à l'aide de l' `userState: suspended` option :

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-ontap-nas
spec:
  version: 1
  backendName: customBackend
  storageDriverName: ontap-nas
  managementLIF: <redacted>
  svm: nas-svm
userState: suspended
  credentials:
    name: backend-tbc-ontap-nas-secret
```

version

Utiliser `version` indicateurs pour imprimer la version de `tridentctl` Et le service exécutant Trident.

```
tridentctl version [flags]
```

Alarmes

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

Prise en charge des plug-ins

Tridentctl prend en charge des plug-ins similaires à kubectl. Tridentctl détecte un plugin si le nom du fichier binaire du plugin suit le schéma "tridentctl-<plugin>" et que le binaire se trouve dans un dossier répertorié dans la variable d'environnement PATH. Tous les plugins détectés sont répertoriés dans la section plugin de l'aide tridentctl. Vous pouvez également limiter la recherche en spécifiant un dossier de plug-ins dans la variable d'environnement TRIDENTCTL_PLUGIN_PATH (exemple : `TRIDENTCTL_PLUGIN_PATH=~/tridentctl-plugins/`). Si la variable est utilisée, tridenctl recherche uniquement dans le dossier spécifié.

Surveillez Trident

Trident fournit un ensemble de terminaux de metrics Prometheus que vous pouvez utiliser pour contrôler les performances d'Trident.

Présentation

Grâce aux mesures fournies par Trident, vous pouvez :

- Surveillez l'état et la configuration de 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. Vous pouvez examiner le temps nécessaire à Trident pour communiquer avec les 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 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. "Blog"Vous apprendrez ainsi à utiliser Prometheus et Grafana avec Trident pour récupérer des metrics. Découvrez sur ce blog comment exécuter

Prometheus en tant qu'opérateur dans votre cluster Kubernetes et comment créer un ServiceMonitor pour obtenir des metrics 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` service et recherche spécifiquement le `metrics` point final du service. Par conséquent, Prometheus est désormais configuré pour comprendre les metrics de Trident.

Outre les mesures directement disponibles auprès de Trident, kubelet expose de nombreuses `kubelet_volume_*` mesures 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. Reportez-vous à la "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 de 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 de REPOS de Trident via 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 Trident**

```
sum by (operation)
(trident_operation_duration_milliseconds_sum{success="true"}) / sum by
(operation)
(trident_operation_duration_milliseconds_count{success="true"})
```

Obtenez des informations sur l'utilisation de 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
```

En savoir plus sur la télémétrie Trident AutoSupport

Par défaut, Trident envoie chaque jour des metrics Prometheus et des informations de base sur le back-end à NetApp.

- Pour empêcher Trident d'envoyer des metrics Prometheus et des informations back-end de base à NetApp, transmettez le `--silence-autosupport` drapeau pendant l'installation de Trident.

- Trident peut également envoyer des journaux de conteneur au support NetApp à la demande via `tridentctl send autosupport`. Vous devrez déclencher Trident pour télécharger ses journaux. Avant de soumettre des journaux, vous devez accepter les fichiers NetApp "politique de confidentialité".
- Sauf mention contraire, Trident récupère les journaux des 24 dernières heures.
- Vous pouvez spécifier la durée de conservation du journal avec l' `--since`indicateur. Par exemple : `tridentctl send autosupport --since=1h`. Ces informations sont collectées et envoyées via un `trident-autosupport` conteneur installé en même temps que Trident. Vous pouvez obtenir l'image du conteneur à l'adresse "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 envoyée par 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`.

Désactivez les mesures 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.

Désinstaller Trident

Vous devez utiliser la même méthode pour désinstaller Trident que celle utilisée pour installer Trident.

Description de la tâche

- Si vous avez besoin d'un correctif pour les bogues observés après une mise à niveau, des problèmes de dépendance ou une mise à niveau non réussie ou incomplète, désinstallez Trident et réinstallez la version précédente en suivant les instructions spécifiques à cette mise à niveau "version". Il s'agit de la seule méthode recommandée pour *rétrograder* vers une version antérieure.
- Pour faciliter la mise à niveau et la réinstallation, la désinstallation de Trident ne supprime pas les CRD ou les objets associés créés par Trident. Si vous devez supprimer complètement Trident et toutes ses données, reportez-vous à la "Retirez complètement les Trident et les CRD".

Avant de commencer

Si vous désaffectez des clusters Kubernetes, vous devez supprimer toutes les applications qui utilisent des volumes créés par Trident avant de procéder à la désinstallation. Cela permet de s'assurer que les ESV ne sont pas publiées sur les nœuds Kubernetes avant d'être supprimées.

Déterminez la méthode d'installation d'origine

Vous devez utiliser la même méthode pour désinstaller Trident que celle utilisée pour l'installer. Avant de procéder à la désinstallation, vérifiez la version que vous avez utilisée pour installer Trident à l'origine.

1. Utiliser `kubectl get pods -n trident` pour examiner les pods.
   - S'il n'y a pas de module opérateur, Trident a été installé à l'aide de `tridentctl`.
   - S'il existe un module opérateur, Trident a été installé à l'aide de l'opérateur Trident soit manuellement, soit à l'aide de l'assistant.
2. S'il y a un module opérateur, utilisez `kubectl describe tproc trident` pour déterminer si Trident a été installé à l'aide de l'assistant.
   - S'il y a une étiquette Helm, Trident a été installé à l'aide de Helm.
   - S'il n'y a pas d'étiquette Helm, Trident a été installé manuellement à l'aide de l'opérateur Trident.

Désinstallez l'installation d'un opérateur Trident

Vous pouvez désinstaller manuellement l'installation d'un opérateur trident ou à l'aide d'Helm.

Désinstallez l'installation manuelle

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

1. **Modifier `TridentOrchestrator` CR et définir l'indicateur de désinstallation** :

```
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.

2. **Supprimer `TridentOrchestrator`** : en supprimant la `TridentOrchestrator` CR utilisée pour déployer Trident, vous demandez à l'opérateur de désinstaller Trident. L'opérateur procède à la suppression du `TridentOrchestrator` déploiement et du démonset Trident, en supprimant les pods Trident qu'il avait créés dans le cadre de l'installation.

```
kubectl delete -f deploy/<bundle.yaml> -n <namespace>
```

Désinstallez l'installation d'Helm

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

```
#List the Helm release corresponding to the 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
```

Désinstallez un `tridentctl` installation

Utilisez la `uninstall` commande dans `tridentctl` pour supprimer toutes les ressources associées à Trident, à l'exception des CRD et des objets associés :

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

Trident pour Docker

Conditions préalables au déploiement

Avant de pouvoir déployer Trident, vous devez installer et configurer les prérequis en matière de protocole sur votre hôte.

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 prérequis de protocole sont installés et configurés sur votre hôte.

Outils NFS

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

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.

Outils iSCSI

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

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
```

Outils NVMe

Installez les outils NVMe à l'aide des commandes correspondant à votre système d'exploitation.

- NVMe requiert RHEL 9 ou version ultérieure.
- Si la version du noyau de votre nœud Kubernetes est trop ancienne ou si le package NVMe n'est pas disponible pour votre version du noyau, vous devrez peut-être mettre à jour la version du noyau de votre nœud avec le package NVMe.

RHEL 9

```
sudo yum install nvme-cli
sudo yum install linux-modules-extra-$(uname -r)
sudo modprobe nvme-tcp
```

Ubuntu

```
sudo apt install nvme-cli
sudo apt -y install linux-modules-extra-$(uname -r)
sudo modprobe nvme-tcp
```

Déployez Trident

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 de Trident peuvent s'exécuter 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". Lorsque vous vous êtes assuré que les conditions préalables sont remplies, vous êtes prêt à déployer Trident.

Méthode de plug-in géré Docker (version 1.13/17.03 et ultérieure)

Avant de commencer

Si vous avez utilisé Trident antérieur à Docker 1.13/17.03 dans la méthode démon classique, veillez à arrêter le processus 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 Trident à l'aide du système de plug-in géré. Remplacez `<version>`-le par la version du plug-in (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 Trident pour consommer du stockage 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/v24.10.0/trident-
installer-24.10.0.tar.gz
tar zxf trident-installer-24.10.0.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 fichier binaire et créé le fichier de configuration, démarrez le démon Trident en utilisant le 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
```

Démarrez 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 de Trident

Vous pouvez mettre à niveau Trident pour Docker en toute sécurité, sans impact sur les volumes utilisés. Au cours du processus de mise à niveau, les `docker volume` commandes dirigées vers le plug-in ne réussiront pas et les applications ne pourront pas monter les volumes tant que le plug-in ne sera pas exécuté à nouveau. Dans la plupart des cas, c'est une question de secondes.

Mise à niveau

Effectuez les étapes ci-dessous pour mettre à niveau 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 de Trident remplace nDVP. Vous devez effectuer une mise à niveau directe de l' `netapp/ndvp-plugin`image vers l' `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 une mise à niveau d'une ancienne version de Trident (antérieure à 20.10) vers Trident 20.10 ou une version ultérieure, vous risquez de générer une erreur. Pour plus d'informations, reportez-vous "Problèmes connus"à . Si vous entrez dans l'erreur, vous devez d'abord désactiver le plug-in, puis supprimer le plug-in, puis installer la version 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

Procédez comme suit pour désinstaller 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 des commandes standard `docker volume` et spécifier le nom du pilote Trident si nécessaire.

Créer un volume

- Créez un volume avec un pilote à l'aide du nom par défaut :

```
docker volume create -d netapp --name firstVolume
```

- Créer un volume avec une instance 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.

Effectuez les étapes ci-dessous pour mettre à niveau Trident pour Docker.

Clonez un volume

Lorsque vous utilisez `ontap-nas` , `ontap-san`, , `solidfire-san` et `gcp-cvs storage drivers`, Trident peut cloner des volumes. Lorsque vous utilisez les `ontap-nas-flexgroup` pilotes ou `ontap-nas-`

`economy`, le clonage 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) par des conteneurs en utilisant Trident **uniquement** s'ils n'ont pas de partitions et si leur système de fichiers est pris en charge par Trident (par exemple, un fichier `ext4` formaté `/dev/sdc1` ne sera pas accessible via 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.
`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 permet de gagner beaucoup de temps mais réduit les économies de stockage. Il est donc préférable de séparer immédiatement le clone.

Option	Description
`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, tout volume provisionné dans Trident est activé. Pour plus d'informations, reportez-vous à la section : "Fonctionnement de 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 MySQL officielle, ne fonctionnent pas comme prévu quand 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`.

Option	Description
`spaceAllocation`	Régler sur `false` Va désactiver la fonctionnalité 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 d'instantanés n'est pas spécifiée et que la règle d'instantanés est `none`, Trident utilise une réserve d'instantanés 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 Trident à l'aide de la méthode de plug-in géré recommandée (à l'aide de `docker plugin` commandes), affichez-les 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 vous trouvez que ce n'est pas suffisant, 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 fichier binaire lui-même sur l'hôte, les journaux sont disponibles dans le système hôte `/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.

Gestion de plusieurs instances Trident

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

Consultez les options de configuration disponibles pour vos configurations Trident.

Options de configuration globale

Ces options de configuration s'appliquent à toutes les configurations Trident, quelle que soit la plateforme de stockage utilisée.

Option	Description	Exemple
`version`	Numéro de version du fichier de configuration	`1`
`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_`

Option	Description	Exemple
`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 non fourni, Trident récupère les LIFs 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. Trident utilise "Mappage de LUN sélectif ONTAP" pour découvrir les LIF iSCI nécessaires à l'établissement d'une session à chemins multiples. 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 au 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`	Trident crée et gère par nœud `igroups` en tant que `netappdvp`. Cette valeur ne peut pas être modifiée ou omise. **Disponible uniquement pour le `ontap-san` pilote**.	`netappdvp`
`limitVolumeSize`	Taille maximale du volume éligible.	`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`
`sanType`	**Pris en charge pour le `ontap-san` pilote uniquement.** Utilisez pour sélectionner `iscsi` pour iSCSI, `nvme` pour NVMe/TCP ou `fcp` pour SCSI over Fibre Channel (FC). **'fcp' (SCSI over FC) est une fonctionnalité de présentation technique dans la version Trident 24.10.**	`iscsi` si vide
`limitVolumePoolSize`	**Pris en charge pour `ontap-san-economy` les `ontap-san-economy` pilotes et uniquement.** Limite la taille des FlexVol pour les modèles économiques ONTAP ONTAP-nas et ONTAP-SAN.	`300g`

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`
`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, tout volume provisionné dans Trident est activé. Pour plus d'informations, reportez-vous à la section : "Fonctionnement de 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 au `.snapshot` répertoire.	« True » pour NFSv4 « false » pour NFSv3
`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 volume unique pouvant 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. Lors du provisionnement d'un volume FlexGroup, Trident utilise tous les agrégats affectés à un SVM. 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 du membre/volume FlexGroup est de 100 Gio.
- Le clonage n'est pas pris en charge pour les volumes FlexGroup.

Pour plus d'informations sur les FlexGroups et les charges de travail appropriées pour FlexGroups, reportez-vous au "Guide des meilleures pratiques et d'implémentation des volumes NetApp FlexGroup" .

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`.

Rôle ONTAP personnalisé pour Trident

Vous pouvez créer un rôle de cluster ONTAP avec une Privileges minimale afin de ne pas avoir à utiliser le rôle ONTAP admin pour effectuer des opérations dans Trident. Lorsque vous incluez le nom d'utilisateur dans une configuration Trident backend, Trident utilise le rôle de cluster ONTAP que vous avez créé pour effectuer les opérations.

Pour plus d'informations sur la création de rôles personnalisés Trident, reportez-vous à la section"Générateur

de rôle personnalisé Trident".

Utilisation de l'interface de ligne de commandes ONTAP

1. Créez un rôle à l'aide de la commande suivante :

```
security login role create <role_name\> -cmddirname "command" -access all
–vserver <svm_name\>
```

2. Créez un nom d'utilisateur pour l'utilisateur Trident :

```
security login create -username <user_name\> -application ontapi
-authmethod <password\> -role <name_of_role_in_step_1\> -vserver
<svm_name\> -comment "user_description"
```

3. Mapper le rôle à l'utilisateur :

```
security login modify username <user_name\> –vserver <svm_name\> -role
<role_name\> -application ontapi -application console -authmethod
<password\>
```

À l'aide de System Manager

Dans ONTAP System Manager, effectuez les opérations suivantes :

1. **Créer un rôle personnalisé** :
   a. Pour créer un rôle personnalisé au niveau du cluster, sélectionnez **Cluster > Paramètres**.

      (Ou) pour créer un rôle personnalisé au niveau du SVM, sélectionner **stockage > Storage VM > >> Paramètres > `required SVM` utilisateurs et rôles**.

   b. Sélectionnez l'icône de flèche (→) en regard de **utilisateurs et rôles**.
   c. Sélectionnez **+Ajouter** sous **rôles**.
   d. Définissez les règles du rôle et cliquez sur **Enregistrer**.
2. **Mapper le rôle à l'utilisateur Trident**: + effectuez les étapes suivantes sur la page **utilisateurs et rôles** :
   a. Sélectionnez Ajouter l'icône **+** sous **utilisateurs**.
   b. Sélectionnez le nom d'utilisateur requis et sélectionnez un rôle dans le menu déroulant pour **role**.
   c. Cliquez sur **Enregistrer**.

Pour plus d'informations, reportez-vous aux pages suivantes :

- "Rôles personnalisés pour l'administration de ONTAP" ou "Définissez des rôles personnalisés"
- "Travaillez avec les rôles et les utilisateurs"

Exemples de fichiers de configuration ONTAP

Exemple NFS pour le <code>ontap-nas</code> 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 le <code>ontap-nas-flexgroup</code> 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 le <code>ontap-nas-economy</code> 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 le <code>ontap-san</code> 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": "netappdvp"
}
```

Exemple NFS pour le <code>ontap-san-economy</code> 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": "netappdvp"
}
```

Exemple NVMe/TCP pour le <code>ontap-san</code> pilote

```
{
  "version": 1,
  "backendName": "NVMeBackend",
  "storageDriverName": "ontap-san",
  "managementLIF": "10.0.0.1",
  "svm": "svm_nvme",
  "username":"vsadmin",
  "password":"password",
  "sanType": "nvme",
  "useREST": true
}
```

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 la version 1.3.2 et effectué une mise à niveau avec des volumes existants, vous devez définir cette valeur pour accéder aux anciens volumes mappés avec 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

Trouvez des informations sur les problèmes connus et les limites lors de l'utilisation de 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 présente certains comportements qui empêchent Trident de le prendre en charge avec toutes les combinaisons de stockage et de pilotes.

- 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 volume (y compris Trident) doivent s'exécuter indépendamment sur chaque nœud d'un cluster Swarm. En raison du fonctionnement de ONTAP et du `ontap-nas` fonctionnement des pilotes et `ontap-san`, ce sont eux qui sont les seuls à pouvoir 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.

Meilleures pratiques et recommandations

Déploiement

Suivez les recommandations répertoriées ici lors du déploiement de Trident.

Déploiement dans un namespace dédié

"Espaces de noms" séparation administrative entre les différentes applications et obstacle au partage des ressources. Par exemple, un volume persistant ne peut pas être consommé depuis un autre espace de noms. Trident fournit des ressources PV à tous les namespaces du cluster Kubernetes et utilise donc un compte de service qui a un Privileges élevé.

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. Indépendamment de la situation, le LIF de gestion doit être accessible via DNS pour faciliter une flexibilité maximale "SVM-DR" Utilisation en association 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. Reportez-vous à la section "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.

Limitez la taille maximale des volumes FlexVol créés par Trident

Pour configurer la taille maximale des volumes FlexVol utilisés comme pools pour les pilotes ONTAP-san-Economy et ONTAP-nas-Economy, utilisez le `limitVolumePoolSize` paramètre dans votre `backend.json` définition.

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.

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 de "évitez un tyran" Ou un conteneur hors contrôle de affectant les charges de travail en dehors du 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 la QoS, reportez-vous à la section "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 de QoS, reportez-vous à la 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'export. 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. Reportez-vous à la section "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"
- "Guide d'administration DU 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 Trident

Pour intégrer Trident, les éléments de conception et d'architecture suivants nécessitent une intégration : sélection et déploiement des pilotes, conception des classes de stockage, conception des pools virtuels, impact de la demande de volume persistant sur le provisionnement du stockage, les opérations des volumes et le déploiement des services OpenShift à l'aide de 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.

Notez que dans les tableaux ci-dessous, toutes les fonctionnalités ne sont pas exposées via 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]

Trident propose 2 pilotes SAN pour ONTAP, dont les fonctionnalités sont indiquées 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 Trident Yes [2]: Géré par Trident, mais non par PV granulaire Yes [3]: Non géré par Trident et non par PV granulaire Yes [4]: Supporté pour les volumes de bloc brut Yes [5]: 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 NetApp ONTAP, vous exploitez les fonctionnalités, les performances et les capacités d'administration d'NetApp que vous connaissez déjà, 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 ONTAP et API d'administration. 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 Trident Yes [2] : pris en charge pour les volumes de bloc brut

Pilotes Azure NetApp Files backend

Trident utilise le `azure-netapp-files` pilote pour gérer le "Azure NetApp Files"service.

Pour plus d'informations sur ce pilote et sur sa configuration"Configuration Trident back-end pour Azure NetApp Files", reportez-vous à la section .

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 Trident

Cloud Volumes Service sur le pilote back-end Google Cloud

Trident utilise le `gcp-cvs` pilote pour établir un lien avec Cloud Volumes Service sur Google Cloud.

Le `gcp-cvs` pilote utilise des pools virtuels pour extraire le back-end et permettre à Trident de déterminer le placement des volumes. L'administrateur définit les pools virtuels dans les `backend.json` fichiers. Les classes de stockage utilisent des sélecteurs pour identifier les pools virtuels par étiquette.

- Si des pools virtuels sont définis en back-end, Trident essaie de créer un volume dans les pools de stockage Google Cloud auxquels ces pools virtuels sont limités.
- Si les pools virtuels ne sont pas définis dans le back-end, Trident sélectionne un pool de stockage Google Cloud dans les pools de stockage disponibles de la région.

Pour configurer le back-end Google Cloud sur Trident, vous devez spécifier `projectNumber`, `apiRegion` et `apiKey` dans le fichier back-end. 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 Cloud Volumes Service sur les types de services et les niveaux de service Google Cloud, reportez-vous à "En savoir plus sur la prise en charge de Trident pour CVS pour GCP"la section .

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 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 n'importe quel attribut spécifié. Le `additionalStoragePools` paramètre permet d'étendre l'ensemble des pools utilisés par Trident pour le provisionnement, ainsi que l'ensemble des pools sélectionnés par les attributs et les `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 Trident back-end. Vous pouvez définir des pools virtuels pour n'importe quel système back-end, à l'aide de n'importe quel pilote fourni par 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, Trident choisit un back-end qui correspond à toutes les étiquettes du sélecteur pour placer le volume. Si les étiquettes du sélecteur de classe de stockage correspondent à plusieurs pools de stockage, Trident choisit l'un d'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. Des pools virtuels peuvent être définis pour tous les systèmes NetApp back-end pris en charge avec 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. Configurer le back-end Azure NetApp Files avec plusieurs étiquettes représentant différents niveaux de performances. 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 Trident lors de la création d'une demande de 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.

Trident tente de faire correspondre le protocole de stockage utilisé avec la méthode d'accès spécifiée selon 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, certains aspects du volume ne peuvent pas être modifiés 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. 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

Trident prend en charge la création de copies Snapshot de volume à la demande et la création d'ESV à partir de copies Snapshot à l'aide du 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

Trident prend également en charge la création de volumes Persistentvolumes à partir de snapshots de volumes. Pour ce faire, il suffit de créer une demande de volume persistant et de mentionner l' `datasource` comme instantané requis à partir duquel le volume doit être créé. Trident traitera cette demande de volume persistant en créant un volume avec les données 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 ni Trident ni le cluster Kubernetes, tant que l'agrégat de destination est un auquel le SVM utilisé par Trident peut accéder. Important : si l'agrégat vient d'être ajouté au SVM, le back-end devra être actualisé en l'ajoutant à Trident. Cela déclenchera Trident à réinventorier le SVM afin que le nouvel agrégat soit reconnu.

Cependant, la migration de volumes entre systèmes back-end n'est pas prise en charge automatiquement par Trident. Cela inclut entre les SVM du même cluster, entre les clusters ou sur une plateforme de stockage différente (même si ce système de stockage est connecté à Trident).

Si un volume est copié vers un autre emplacement, la fonctionnalité d'importation de volume peut être utilisée pour importer les volumes actuels dans Trident.

Développement des volumes

Trident prend en charge le redimensionnement des volumes persistants NFS et iSCSI. 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 autoriser une éventuelle extension ultérieurement, définissez `allowVolumeExpansion` sur `true` dans votre classe de stockage associée au volume. Lorsque le volume persistant doit être redimensionné, modifiez l' `spec.resources.requests.storage`annotation de la demande de volume persistant en fonction de 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` les pilotes, utilisez la commande `tridentctl import volume <backend-name> <volume-name> -f /path/pvc.yaml` pour importer un volume existant dans Kubernetes qui sera géré par Trident. Le fichier ESV YAML ou JSON utilisé dans la commande de volume d'importation pointe vers une classe de stockage qui identifie 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` pilote ou `gcp-cvs` est utilisé, utilisez la commande `tridentctl import volume <backend-name> <volume path> -f /path/pvc.yaml` pour importer le volume dans Kubernetes et le gérer par Trident. Cela garantit une référence de volume unique.

Lors de l'exécution de la commande ci-dessus, Trident trouve le volume sur le back-end et lit sa taille. Il ajoute automatiquement (et écrase si nécessaire) la taille du volume de la demande de volume configurée. Trident crée ensuite le nouveau volume persistant et Kubernetes lie la demande de 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 ONTAP ne présente pas ces problèmes et peut facilement soutenir un déploiement 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.

Il est important de configurer les variables Ansible selon la méthode de déploiement 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 en ce qui concerne 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, tout comme le service de journalisation, le service de metrics peut être déployé lors de la configuration initiale du cluster ou après son fonctionnement à l'aide de la méthode d'installation des composants. 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.
`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.

Variable	Détails
`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 à l'aide des playbooks des composants après l'installation d'OpenShift, Ansible crée les demandes PVCS requises et, une fois que 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 et reprise d'activité

En savoir plus sur les options de protection et de restauration pour Trident et les volumes créés à l'aide de Trident. Vous devez disposer d'une stratégie de protection et de restauration des données pour chaque application ayant des exigences de persistance.

Réplication et restauration Trident

En cas d'incident, vous pouvez créer une sauvegarde pour restaurer Trident.

Réplication Trident

Trident utilise des CRD Kubernetes pour stocker et gérer son propre état ainsi que celui du cluster Kubernetes pour stocker ses métadonnées.

Étapes

1. Sauvegardez le cluster Kubernetes avec "Kubernetes : sauvegarde d'un cluster ETCD".
2. Placez les artéfacts de sauvegarde sur une FlexVol.

> Nous vous recommandons de protéger la SVM où réside la FlexVol avec une relation SnapMirror vers une autre SVM.

Restauration Trident

Avec les CRD Kubernetes et le snapshot de type ETCD du cluster Kubernetes, vous pouvez restaurer Trident.

Étapes

1. Depuis le SVM de destination, monter le volume qui contient les fichiers de données et les certificats Kubernetes sur l'hôte qui sera configuré en tant que nœud maître.
2. Copiez tous les certificats requis en rapport avec le cluster Kubernetes sous `/etc/kubernetes/pki` et les fichiers membres etcd sous `/var/lib/etcd`.
3. Restaurez le cluster Kubernetes à partir de la sauvegarde ETCD à l'aide de "Kubernetes : restauration d'un cluster ETCD".

4. Courez `kubectl get crd` Pour vérifier que toutes les ressources personnalisées Trident sont disponibles et récupérer les objets Trident afin de vérifier que toutes les données sont disponibles.

Réplication et restauration des SVM

Trident ne peut pas configurer les relations de réplication. Toutefois, l'administrateur du stockage peut utiliser "SnapMirror ONTAP" pour répliquer un SVM.

En cas d'incident, vous pouvez activer la SVM de destination SnapMirror pour démarrer le service des données. Vous pouvez revenir au système principal lorsque les systèmes sont restaurés.

Description de la tâche

Tenir compte des points suivants lors de l'utilisation de la fonction de réplication SVM SnapMirror :

- Vous devez créer un back-end distinct pour chaque SVM lorsque la fonction SVM-DR est activée.
- Configurez les classes de stockage pour sélectionner les systèmes back-end répliqués uniquement en cas de besoin, afin d'éviter que des volumes ne nécessitant pas de réplication provisionnée vers les systèmes back-end qui prennent en charge la SVM-DR.
- Les administrateurs d'applications doivent comprendre les coûts et la complexité supplémentaires associés à la réplication et tenir compte de leur plan de reprise avant de commencer ce processus.

Réplication SVM

Vous pouvez utiliser "ONTAP : réplication SVM SnapMirror" Pour créer la relation de réplication de SVM.

SnapMirror vous permet de définir des options pour contrôler ce qui doit être répliqué. Vous devez savoir quelles options vous avez sélectionnées lors de la préformation Restauration des SVM à l'aide de Trident.

- "-identité-préserver vrai" Réplique l'ensemble de la configuration du SVM.
- "-discard-configs réseau" Exclut les LIFs et les paramètres réseau associés.
- "-identity-preserve false" réplique uniquement les volumes et la configuration de sécurité.

Restauration des SVM à l'aide de Trident

Trident ne détecte pas automatiquement les défaillances des SVM. En cas d'incident, l'administrateur peut initier manuellement le basculement de Trident vers le nouveau SVM.

Étapes

1. Annuler les transferts SnapMirror planifiés et en cours, rompre la relation de réplication, arrêter la SVM source, puis activer la SVM de destination SnapMirror.
2. Si vous avez spécifié `-identity-preserve false` ou `-discard-config network` Lors de la configuration de la réplication de votre SVM, mettre à jour `managementLIF` et `dataLIF` Dans le fichier de définition du back-end Trident.
3. Confirmer `storagePrefix` Est présent dans le fichier de définition du back-end Trident. Ce paramètre ne peut pas être modifié. Omission `storagePrefix` provoque l'échec de la mise à jour du back-end.
4. Mettre à jour tous les systèmes back-end nécessaires pour indiquer le nom du nouveau SVM de destination à l'aide de :

```
./tridentctl update backend <backend-name> -f <backend-json-file> -n
<namespace>
```

5. Si vous avez spécifié `-identity-preserve false` ou `discard-config network`, vous devez rebondir tous les pods d'application.

> Si vous avez spécifié `-identity-preserve true`, tous les volumes provisionnés par Trident commencent à transmettre des données lorsque le SVM de destination est activé.

Réplication et restauration de volume

Trident ne peut pas configurer les relations de réplication SnapMirror. Toutefois, l'administrateur du stockage peut utiliser "Réplication et restauration ONTAP SnapMirror" pour répliquer les volumes créés par Trident.

Vous pouvez ensuite importer les volumes récupérés dans Trident à l'aide de "importation de volume tridentctl".

> L'importation n'est pas prise en charge sur `ontap-nas-economy`, `ontap-san-economy`, ou `ontap-flexgroup-economy` pilotes.

Protection des données Snapshot

Vous pouvez protéger et restaurer les données à l'aide des éléments suivants :

- Un contrôleur de snapshot externe et des CRD pour créer des copies Snapshot de volume Kubernetes de volumes persistants (PVS).

  "Snapshots de volume"

- Snapshots ONTAP pour restaurer le contenu complet d'un volume ou pour restaurer des fichiers individuels ou des LUN.

  "Snapshots ONTAP"

Sécurité

Sécurité

Suivez les recommandations ci-dessous pour vous assurer que votre installation Trident est sécurisée.

Exécutez 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 Trident ou aux pods afin d'assurer un stockage fiable et de bloquer les activités malveillantes potentielles.

Pour séparer les autres applications et utilisateurs de Trident, installez toujours Trident dans son propre

espace de noms Kubernetes (`trident`). Le placement de Trident dans son propre espace de noms permet de garantir que seules les équipes d'administration Kubernetes ont accès au pod Trident et aux artéfacts (comme les secrets back-end et CHAP, le cas échéant) stockés dans les objets CRD dont le nom a été donné. Vous devez vous assurer d'autoriser uniquement les administrateurs à accéder à l'espace de noms Trident et donc à l' `tridentctl`application.

Utilisez l'authentification CHAP avec les systèmes back-end ONTAP SAN

Trident prend en charge l'authentification CHAP pour les charges de travail SAN ONTAP (à l'aide `ontap-san` de pilotes et `ontap-san-economy`). NetApp recommande l'utilisation du protocole CHAP bidirectionnel avec Trident pour l'authentification entre un hôte et le back-end de stockage.

Pour les systèmes back-end ONTAP qui utilisent des pilotes de stockage SAN, Trident peut configurer le protocole CHAP bidirectionnel et gérer les noms d'utilisateur et les secrets CHAP via `tridentctl`. Reportez-vous à la section "Préparez la configuration du système back-end avec les pilotes SAN ONTAP" pour savoir comment Trident configure CHAP sur des systèmes back-end ONTAP.

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. Trident utilise un objet secret qui inclut deux mots de passe CHAP par locataire. Lorsque Trident est installé, il gère les secrets CHAP et les stocke dans un `tridentvolume` objet CR pour le PV correspondant. Lorsque vous créez un volume persistant, Trident utilise les secrets CHAP pour lancer une session iSCSI et communiquer avec le système NetApp HCI et SolidFire via CHAP.

Les volumes créés par Trident ne sont associés à aucun groupe d'accès de volume.

Utilisez 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, tout volume provisionné dans Trident est activé.
- Si NAE n'est pas activé sur le back-end, tout volume provisionné dans Trident est activé sur NVE, à moins que vous ne définiez l'indicateur de chiffrement NVE sur `false` dans la configuration back-end.

Les volumes créés dans Trident sur un système back-end 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.
- La définition de l'indicateur de chiffrement NVE `false` sur un système back-end compatible NAE crée un volume compatible NAE. Vous ne pouvez pas désactiver le chiffrement NAE en configurant l'indicateur de chiffrement NVE sur `false`.

- Vous pouvez créer manuellement un volume NVE dans 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 Linux Unified Key Setup (LUKS) pour chiffrer les volumes ONTAP SAN et ONTAP SAN ECONOMY sur Trident. Trident prend en charge la rotation de phrase de passe et l'extension de volume pour les volumes chiffrés LUKS.

Dans Trident, les volumes chiffrés LUKS utilisent le cypher et le mode aes-xts-mclair 64, 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: csi.trident.netapp.io
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.

Configuration back-end pour l'importation de volumes LUKS

Pour importer un volume LUKS, vous devez définir `luksEncryption` sur sur(`true` le back-end. L' `luksEncryption`option indique à Trident si le volume est conforme LUKS (`true`) ou non conforme LUKS) (`false`comme indiqué dans l'exemple suivant.

```
version: 1
storageDriverName: ontap-san
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
svm: trident_svm
username: admin
password: password
defaults:
  luksEncryption: 'true'
  spaceAllocation: 'false'
  snapshotPolicy: default
  snapshotReserve: '10'
```

Configuration PVC pour l'importation de volumes LUKS

Pour importer des volumes LUKS de façon dynamique, définissez l'annotation `trident.netapp.io/luksEncryption` sur `true` et incluez une classe de stockage LUKS activée dans la demande de volume virtuel, comme indiqué dans cet exemple.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: luks-pvc
  namespace: trident
  annotations:
    trident.netapp.io/luksEncryption: "true"
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: luks-sc
```

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. Lorsqu'un nouveau pod est créé, 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: csi.trident.netapp.io
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.

Protégez les applications avec Trident Protect

Découvrez Trident Protect

NetApp Trident Protect propose des fonctionnalités avancées de gestion des données applicatives qui améliorent la fonctionnalité et la disponibilité des applications Kubernetes avec état reposant sur les systèmes de stockage NetApp ONTAP et le mécanisme de provisionnement du stockage NetApp Trident CSI. Trident Protect simplifie la gestion, la protection et le déplacement des workloads conteneurisés dans les clouds publics et les environnements sur site. Il propose également des fonctionnalités d'automatisation via son API et son interface de ligne de commande.

Vous pouvez protéger les applications avec Trident Protect en créant des ressources personnalisées (CRS) ou en utilisant l'interface de ligne de commande Trident Protect.

Et la suite ?

Vous pouvez en savoir plus sur les exigences de Trident Protect avant de l'installer :

- "Exigences de Trident Protect"

Installez Trident Protect

Exigences de Trident Protect

Commencez par vérifier que votre environnement opérationnel, vos clusters d'applications, vos applications et vos licences sont prêts. Déploiement et exploitation de Trident Protect répondent à ces exigences.

Compatibilité avec Trident Protect Kubernetes

Trident Protect est compatible avec un large éventail d'offres Kubernetes entièrement gérées et autogérées, notamment :

- Amazon Elastic Kubernetes Service (EKS)
- Google Kubernetes Engine (GKE)
- Microsoft Azure Kubernetes Service (AKS)
- Red Hat OpenShift
- SUSE Rancher
- Gamme VMware Tanzu
- Kubernetes en amont

Trident protège la compatibilité back-end du stockage

Trident Protect prend en charge les systèmes back-end suivants :

- Amazon FSX pour NetApp ONTAP

- Cloud Volumes ONTAP
- Baies de stockage ONTAP
- Google Cloud NetApp volumes
- Azure NetApp Files

Assurez-vous que votre système back-end répond aux exigences suivantes :

- Assurez-vous que le stockage NetApp connecté au cluster utilise Astra Trident 24.02 ou version ultérieure (Trident 24.10 est recommandé).
  - Si Astra Trident est antérieure à la version 24.06.1 et que vous prévoyez d'utiliser la fonctionnalité de reprise d'activité NetApp SnapMirror, vous devez activer manuellement Astra Control provisionner.
- Vérifiez que vous disposez de la dernière version d'Astra Control Provisioner (installée et activée par défaut à partir d'Astra Trident 24.06.1).
- Assurez-vous de disposer d'un système back-end de stockage NetApp ONTAP.
- Assurez-vous d'avoir configuré un compartiment de stockage objet pour le stockage des sauvegardes.
- Créez tous les espaces de noms d'applications que vous prévoyez d'utiliser pour les opérations de gestion des données d'applications ou d'applications. Trident Protect ne crée pas ces espaces de noms pour vous ; si vous spécifiez un espace de noms inexistant dans une ressource personnalisée, l'opération échoue.

Conditions requises pour les volumes d'économie nas

Trident Protect prend en charge les opérations de sauvegarde et de restauration sur les volumes économiques nas. Les copies Snapshot, les clones et la réplication SnapMirror sur des volumes économiques nas ne sont pas pris en charge actuellement. Vous devez activer un répertoire d'instantanés pour chaque volume d'économie nas que vous prévoyez d'utiliser avec Trident Protect.

Certaines applications ne sont pas compatibles avec les volumes qui utilisent un répertoire de snapshots. Pour ces applications, vous devez masquer le répertoire des snapshots en exécutant la commande suivante sur le système de stockage ONTAP :

```
nfs modify -vserver <svm> -v3-hide-snapshot enabled
```

Vous pouvez activer le répertoire des snapshots en exécutant la commande suivante pour chaque volume nas-Economy, en remplaçant `<volume-UUID>` par l'UUID du volume à modifier :

```
tridentctl update volume <volume-UUID> --snapshot-dir=true --pool-level
=true -n trident
```

Vous pouvez activer les répertoires de snapshots par défaut pour les nouveaux volumes en définissant l'option de configuration du back-end Trident `snapshotDir` sur `true`. Les volumes existants ne sont pas affectés.

Conditions requises pour la réplication SnapMirror

NetApp SnapMirror est disponible pour les solutions ONTAP suivantes avec Trident Protect :

- NetApp ASA
- NetApp AFF
- NetApp FAS
- NetApp ONTAP Select
- NetApp Cloud Volumes ONTAP
- Amazon FSX pour NetApp ONTAP

Configuration requise pour un cluster ONTAP pour la réplication SnapMirror

Si vous prévoyez d'utiliser la réplication SnapMirror, assurez-vous que votre cluster ONTAP répond aux exigences suivantes :

- **Astra Control Provisioner ou Trident** : Astra Control Provisioner ou Trident doit exister sur les clusters Kubernetes source et de destination qui utilisent ONTAP en tant que back-end. Trident Protect prend en charge la réplication avec la technologie NetApp SnapMirror au moyen de classes de stockage basées sur les pilotes suivants :
  - `ontap-nas`
  - `ontap-san`
- **Licences** : les licences asynchrones de SnapMirror ONTAP utilisant le bundle protection des données doivent être activées sur les clusters ONTAP source et cible. Pour plus d'informations, reportez-vous à la section "Présentation des licences SnapMirror dans ONTAP" .

Considérations de peering pour la réplication SnapMirror

Si vous prévoyez d'utiliser le peering back-end, assurez-vous que votre environnement répond aux exigences suivantes :

- **Cluster et SVM** : les systèmes back-end de stockage ONTAP doivent être peering. Pour plus d'informations, reportez-vous à la section "Présentation du cluster et de SVM peering" .

  S'assurer que les noms de SVM utilisés dans la relation de réplication entre deux clusters ONTAP sont uniques.

- **Astra Control Provisioner ou Trident et SVM** : les SVM distants à peering doivent être disponibles pour Astra Control Provisioner ou Trident sur le cluster destination.
- **Systèmes back-end gérés** : vous devez ajouter et gérer des systèmes back-end de stockage ONTAP dans Trident Protect pour créer une relation de réplication.
- **NVMe over TCP** : Trident Protect ne prend pas en charge la réplication NetApp SnapMirror pour les systèmes back-end de stockage qui utilisent le protocole NVMe over TCP.

Configuration Trident/ONTAP pour la réplication SnapMirror

Trident Protect exige que vous configuriez au moins un système back-end de stockage qui prend en charge la réplication à la fois pour les clusters source et de destination. Si les clusters source et cible sont identiques, l'application de destination doit utiliser un back-end de stockage différent de l'application source pour une résilience optimale.

Facteurs à prendre en compte lors de l'utilisation de KubeVirt

Si vous prévoyez d'utiliser des "KubeVirt" machines virtuelles avec la réplication SnapMirror, vous devez configurer la virtualisation pour pouvoir geler et annuler le gel de vos SVM. Après avoir configuré la virtualisation, les SVM que vous déployez incluent les outils nécessaires pour geler et débloquer. Pour en savoir plus sur la configuration de la virtualisation, reportez-vous à "Installation d'OpenShift Virtualization"la section .

Installer et configurer Trident Protect

Si votre environnement satisfait aux exigences de Trident Protect, vous pouvez suivre ces étapes pour installer Trident Protect sur votre cluster. Vous pouvez obtenir Trident Protect de NetApp ou l'installer à partir de votre propre registre privé. L'installation à partir d'un registre privé est utile si votre cluster ne peut pas accéder à Internet.

> Par défaut, Trident Protect collecte des informations sur le support que vous pouvez ouvrir dans tous les dossiers de support NetApp, y compris les journaux, les metrics et les informations de topologie des clusters et des applications gérées. Trident Protect envoie ces offres de support à NetApp selon un calendrier quotidien. Vous pouvez éventuellement désactiver cette collection de packs de prise en charge lorsque vous installez Trident Protect. Vous pouvez le faire manuellement "générer un bundle de support"à tout moment.

Installez Trident Protect from NetApp

1. Ajout du référentiel Trident Helm :

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

2. Installez les CRD Trident Protect :

```
helm install trident-protect-crds netapp-trident-protect/trident-
protect-crds --version 100.2410.0 --create-namespace --namespace
trident-protect
```

3. Utilisez Helm pour installer Trident Protect à l'aide de l'une des commandes suivantes. Remplacer `<name_of_cluster>` par un nom de cluster, qui sera attribué au cluster et utilisé pour identifier les sauvegardes et snapshots du cluster :
   - Installez Trident Protect normalement :

```
helm install trident-protect netapp-trident-protect/trident-
protect --set clusterName=<name_of_cluster> --version 100.2410.0
--create-namespace --namespace trident-protect
```

   - Installez Trident Protect et désactivez les téléchargements quotidiens de packs de prise en charge Trident Protect AutoSupport :

```
helm install trident-protect netapp-trident-protect/trident-
protect --set autoSupport.enabled=false --set
clusterName=<name_of_cluster> --version 100.2410.0 --create
-namespace --namespace trident-protect
```

4. Éventuellement, geler vos machines virtuelles. Si vous utilisez la prise en charge de KubeVirt pour SnapMirror, la congélation des VM vous aide à les gérer efficacement :

```
kubectl set env deployment/trident-protect-controller-manager
NEPTUNE_VM_FREEZE=true -n trident-protect
```

> Vous devez configurer la virtualisation pour que la fonctionnalité de blocage fonctionne. Les machines virtuelles déployées après cette configuration incluent les binaires nécessaires au gel et au dégel. Pour en savoir plus sur la configuration de la virtualisation, reportez-vous à "Installation d'OpenShift Virtualization"la section .

Installez Trident Protect à partir d'un registre privé

Vous pouvez installer Trident Protect à partir d'un registre d'images privé si votre cluster Kubernetes ne peut pas accéder à Internet. Dans ces exemples, remplacez les valeurs entre parenthèses par les informations de votre environnement :

1. Extrayez les images suivantes sur votre ordinateur local, mettez à jour les balises, puis envoyez-les vers votre registre privé :

```
netapp/controller:24.10.0
netapp/restic:24.10.0
netapp/kopia:24.10.0
netapp/trident-autosupport:24.10.0
netapp/exechook:24.10.0
netapp/resourcebackup:24.10.0
netapp/resourcerestore:24.10.0
netapp/resourcedelete:24.10.0
bitnami/kubectl:1.30.2
kubebuilder/kube-rbac-proxy:v0.16.0
```

   Par exemple :

```
docker pull netapp/controller:24.10.0
```

```
docker tag netapp/controller:24.10.0 <private-registry-
url>/controller:24.10.0
```

```
docker push <private-registry-url>/controller:24.10.0
```

2. Créer l'espace de noms du système Trident Protect :

```
kubectl create ns trident-protect
```

3. Connectez-vous au registre :

```
helm registry login <private-registry-url> -u <account-id> -p <api-
token>
```

4. Créez un secret Pull à utiliser pour l'authentification de registre privé :

```
kubectl create secret docker-registry regcred --docker
-username=<registry-username> --docker-password=<api-token> -n
trident-protect --docker-server=<private-registry-url>
```

5. Ajout du référentiel Trident Helm :

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

6. Créez un fichier nommé `protectValues.yaml` contenant les paramètres Trident Protect suivants :

```
image:
  registry: <private-registry-url>
imagePullSecrets:
- name: regcred
controller:
  image:
    registry: <private-registry-url>
rbacProxy:
  image:
    registry: <private-registry-url>
crCleanup:
  imagePullSecrets:
  - name: regcred
webhooksCleanup:
  imagePullSecrets:
  - name: regcred
```

7. Installez les CRD Trident Protect :

```
helm install trident-protect-crds netapp-trident-protect/trident-
protect-crds --version 100.2410.0 --create-namespace --namespace
trident-protect
```

8. Utilisez Helm pour installer Trident Protect à l'aide de l'une des commandes suivantes. Remplacer `<name_of_cluster>` par un nom de cluster, qui sera attribué au cluster et utilisé pour identifier les sauvegardes et snapshots du cluster :
   - Installez Trident Protect normalement :

```
helm install trident-protect netapp-trident-protect/trident-
protect --set clusterName=<name_of_cluster> --version 100.2410.0
--create-namespace --namespace trident-protect -f
protectValues.yaml
```

- Installez Trident Protect et désactivez les téléchargements quotidiens de packs de prise en charge Trident Protect AutoSupport :

```
helm install trident-protect netapp-trident-protect/trident-
protect --set autoSupport.enabled=false --set
clusterName=<name_of_cluster> --version 100.2410.0 --create
-namespace --namespace trident-protect -f protectValues.yaml
```

9. Éventuellement, geler vos machines virtuelles. Si vous utilisez la prise en charge de KubeVirt pour SnapMirror, la congélation des VM vous aide à les gérer efficacement :

```
kubectl set env deployment/trident-protect-controller-manager
NEPTUNE_VM_FREEZE=true -n trident-protect
```

Vous devez configurer la virtualisation pour que la fonctionnalité de blocage fonctionne. Les machines virtuelles déployées après cette configuration incluent les binaires nécessaires au gel et au dégel. Pour en savoir plus sur la configuration de la virtualisation, reportez-vous à "Installation d'OpenShift Virtualization"la section .

Installez le plug-in Trident Protect CLI

Vous pouvez utiliser le plug-in de ligne de commande Trident Protect, qui est une extension de l'utilitaire Trident `tridentctl`, pour créer et interagir avec les ressources personnalisées Trident Protect (CRS).

Installez le plug-in Trident Protect CLI

Avant d'utiliser l'utilitaire de ligne de commande, vous devez l'installer sur la machine que vous utilisez pour accéder à votre cluster. Procédez comme suit, selon si votre ordinateur utilise un processeur x64 ou ARM.

Télécharger le plug-in pour les processeurs Linux AMD64

1. Téléchargez le plug-in Trident Protect CLI :

```
curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-protect/releases/download/24.10.0/tridentctl-protect-linux-amd64
```

Télécharger le plug-in pour les processeurs Linux ARM64

1. Téléchargez le plug-in Trident Protect CLI :

```
curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-protect/releases/download/24.10.0/tridentctl-protect-linux-arm64
```

Télécharger le plug-in pour les processeurs Mac AMD64

1. Téléchargez le plug-in Trident Protect CLI :

```
curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-protect/releases/download/24.10.0/tridentctl-protect-macos-amd64
```

Télécharger le plug-in pour les processeurs Mac ARM64

1. Téléchargez le plug-in Trident Protect CLI :

```
curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-protect/releases/download/24.10.0/tridentctl-protect-macos-arm64
```

1. Activer les autorisations d'exécution pour le binaire :

```
chmod +x tridentctl-protect
```

2. Copiez le fichier binaire du plug-in à un emplacement défini dans votre variable PATH. Par exemple, `/usr/bin` ou `/usr/local/bin` (vous pouvez avoir besoin d'un Privileges élevé) :

```
cp ./tridentctl-protect /usr/local/bin/
```

3. Vous pouvez également copier le fichier binaire dans un emplacement de votre répertoire personnel. Dans ce cas, vous devrez peut-être ajouter l'emplacement à votre variable PATH :

```
cp ./tridentctl-protect ~/bin/
```

Afficher l'Trident aide du plug-in de l'interface de ligne

Vous pouvez utiliser les fonctions d'aide du plug-in intégré pour obtenir une aide détaillée sur les fonctionnalités du plug-in :

Étapes

1. Utilisez la fonction d'aide pour afficher les conseils d'utilisation :

```
tridentctl protect help
```

Activer la saisie semi-automatique de la commande

Une fois que vous avez installé le plug-in de l'interface de ligne de commande Trident Protect, vous pouvez activer l'exécution automatique pour certaines commandes.

Activer la saisie semi-automatique pour le shell Bash

1. Téléchargez le script d'achèvement :

```
curl -L -O https://github.com/NetApp/tridentctl-protect/releases/download/24.10.0/tridentctl-completion.bash
```

2. Créez un nouveau répertoire dans votre répertoire personnel pour contenir le script :

```
mkdir -p ~/.bash/completions
```

3. Déplacez le script téléchargé dans le `~/.bash/completions` répertoire :

```
mv tridentctl-completion.bash ~/.bash/completions/
```

4. Ajoutez la ligne suivante au `~/.bashrc` fichier de votre répertoire personnel :

```
source ~/.bash/completions/tridentctl-completion.bash
```

Activer la saisie semi-automatique pour la coque Z.

1. Téléchargez le script d'achèvement :

```
curl -L -O https://github.com/NetApp/tridentctl-protect/releases/download/24.10.0/tridentctl-completion.zsh
```

2. Créez un nouveau répertoire dans votre répertoire personnel pour contenir le script :

```
mkdir -p ~/.zsh/completions
```

3. Déplacez le script téléchargé dans le `~/.zsh/completions` répertoire :

```
mv tridentctl-completion.zsh ~/.zsh/completions/
```

4. Ajoutez la ligne suivante au `~/.zprofile` fichier de votre répertoire personnel :

```
source ~/.zsh/completions/tridentctl-completion.zsh
```

Résultat

Lors de votre prochaine connexion au shell, vous pouvez utiliser la saisie semi-automatique de la commande avec le plugin tridentctl Protect.

Gérez Trident Protect

Gestion des autorisations et du contrôle d'accès

Trident Protect utilise le modèle Kubernetes de contrôle d'accès basé sur des rôles (RBAC). Par défaut, Trident Protect fournit un espace de noms système unique et le compte de service par défaut qui lui est associé. Si vous avez une entreprise avec de nombreux utilisateurs ou des besoins de sécurité spécifiques, vous pouvez utiliser les fonctionnalités RBAC de Trident Protect pour bénéficier d'un contrôle plus granulaire sur l'accès aux ressources et aux espaces de noms.

L'administrateur du cluster a toujours accès aux ressources de l'espace de noms par défaut `trident-protect` et peut également accéder aux ressources de tous les autres espaces de noms. Pour contrôler l'accès aux ressources et aux applications, vous devez créer des espaces de noms supplémentaires et ajouter des ressources et des applications à ces espaces de noms.

Notez qu'aucun utilisateur ne peut créer de CRS de gestion des données d'application dans l'espace de noms par défaut `trident-protect`. Vous devez créer une CRS de gestion des données d'application dans un espace de noms d'application (pour cela, il est recommandé de créer une CRS de gestion des données d'application dans le même espace de nom que l'application associée).

> Seuls les administrateurs doivent avoir accès aux objets de ressources personnalisés Privileged Trident Protect, notamment :
>
> - **AppVault** : nécessite les données d'informations d'identification du compartiment
> - **AutoSupportBundle** : collecte des mesures, des journaux et d'autres données sensibles de Trident Protect
> - **AutoSupportBundleSchedule** : gère les plannings de collecte de journaux
>
> Comme bonne pratique, utilisez RBAC pour limiter l'accès aux objets privilégiés aux administrateurs.

Pour plus d'informations sur la façon dont RBAC réglemente l'accès aux ressources et aux espaces de noms, reportez-vous à la section "Documentation Kubernetes RBAC" .

Pour plus d'informations sur les comptes de service, reportez-vous au "Documentation du compte de service Kubernetes".

Exemple : gestion de l'accès pour deux groupes d'utilisateurs

Par exemple, une organisation dispose d'un administrateur de cluster, d'un groupe d'utilisateurs techniques et d'un groupe d'utilisateurs marketing. L'administrateur du cluster doit effectuer les tâches suivantes pour créer un environnement dans lequel le groupe d'ingénierie et le groupe marketing ont chacun accès uniquement aux ressources affectées à leurs espaces de noms respectifs.

Étape 1 : créez un espace de noms pour contenir des ressources pour chaque groupe

La création d'un espace de noms vous permet de séparer logiquement les ressources et de mieux contrôler qui a accès à ces ressources.

Étapes

1. Créer un espace de nom pour le groupe d'ingénierie :

```
kubectl create ns engineering-ns
```

2. Créez un espace de nom pour le groupe marketing :

```
kubectl create ns marketing-ns
```

Étape 2 : créez de nouveaux comptes de service pour interagir avec les ressources de chaque espace de noms

Chaque nouvel espace de noms que vous créez est fourni avec un compte de service par défaut, mais vous devez créer un compte de service pour chaque groupe d'utilisateurs afin de pouvoir diviser davantage Privileges entre les groupes si nécessaire.

Étapes

1. Créer un compte de service pour le groupe d'ingénierie :

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: eng-user
  namespace: engineering-ns
```

2. Créez un compte de service pour le groupe marketing :

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: mkt-user
  namespace: marketing-ns
```

Étape 3 : créez un secret pour chaque nouveau compte de service

Un secret de compte de service est utilisé pour s'authentifier auprès du compte de service et peut facilement être supprimé et recréé si compromis.

Étapes

1. Créez un secret pour le compte de service d'ingénierie :

```
apiVersion: v1
kind: Secret
metadata:
  annotations:
    kubernetes.io/service-account.name: eng-user
  name: eng-user-secret
  namespace: engineering-ns
type: kubernetes.io/service-account-token
```

2. Créez un secret pour le compte de service marketing :

```
apiVersion: v1
kind: Secret
metadata:
  annotations:
    kubernetes.io/service-account.name: mkt-user
  name: mkt-user-secret
  namespace: marketing-ns
type: kubernetes.io/service-account-token
```

Étape 4 : créez un objet RoleBinding pour lier l'objet ClusterRole à chaque nouveau compte de service

Un objet ClusterRole par défaut est créé lorsque vous installez Trident Protect. Vous pouvez lier ce ClusterRole au compte de service en créant et en appliquant un objet RoleBinding.

Étapes

1. Liez ClusterRole au compte de service d'ingénierie :

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: engineering-ns-tenant-rolebinding
  namespace: engineering-ns
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: trident-protect-tenant-cluster-role
subjects:
- kind: ServiceAccount
  name: eng-user
  namespace: engineering-ns
```

2. Associez ClusterRole au compte de service marketing :

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: marketing-ns-tenant-rolebinding
  namespace: marketing-ns
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: trident-protect-tenant-cluster-role
subjects:
- kind: ServiceAccount
  name: mkt-user
  namespace: marketing-ns
```

Étape 5 : autorisations de test

Vérifiez que les autorisations sont correctes.

Étapes

1. Vérifier que les utilisateurs d'ingénierie peuvent accéder aux ressources d'ingénierie :

```
kubectl auth can-i --as=system:serviceaccount:engineering-ns:eng-user
get applications.protect.trident.netapp.io -n engineering-ns
```

2. Vérifiez que les utilisateurs d'ingénierie ne peuvent pas accéder aux ressources marketing :

```
kubectl auth can-i --as=system:serviceaccount:engineering-ns:eng-user
get applications.protect.trident.netapp.io -n marketing-ns
```

Étape 6 : accorder l'accès aux objets AppVault

Pour effectuer des tâches de gestion des données telles que les sauvegardes et les snapshots, l'administrateur du cluster doit accorder l'accès aux objets AppVault à des utilisateurs individuels.

Étapes

1. Créez et appliquez un fichier YAML de combinaison AppVault et secret qui accorde à un utilisateur l'accès à un AppVault. Par exemple, la CR suivante accorde l'accès à un AppVault à l'utilisateur `eng-user`:

```
apiVersion: v1
data:
  accessKeyID: <ID_value>
  secretAccessKey: <key_value>
kind: Secret
metadata:
  name: appvault-for-eng-user-only-secret
  namespace: trident-protect
type: Opaque
---
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: appvault-for-eng-user-only
  namespace: trident-protect # Trident protect system namespace
spec:
  providerConfig:
    azure:
      accountName: ""
      bucketName: ""
      endpoint: ""
    gcp:
      bucketName: ""
      projectID: ""
    s3:
      bucketName: testbucket
      endpoint: 192.168.0.1:30000
      secure: "false"
      skipCertValidation: "true"
  providerCredentials:
    accessKeyID:
      valueFromSecret:
        key: accessKeyID
        name: appvault-for-eng-user-only-secret
    secretAccessKey:
      valueFromSecret:
        key: secretAccessKey
        name: appvault-for-eng-user-only-secret
  providerType: GenericS3
```

2. Créez et appliquez une CR de rôle pour permettre aux administrateurs de cluster d'accorder l'accès à des ressources spécifiques dans un espace de noms. Par exemple :

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: eng-user-appvault-reader
  namespace: trident-protect
rules:
- apiGroups:
  - protect.trident.netapp.io
  resourceNames:
  - appvault-for-enguser-only
  resources:
  - appvaults
  verbs:
  - get
```

3. Créez et appliquez une CR RoleBinding pour lier les autorisations à l'utilisateur eng-user. Par exemple :

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: eng-user-read-appvault-binding
  namespace: trident-protect
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: eng-user-appvault-reader
subjects:
- kind: ServiceAccount
  name: eng-user
  namespace: engineering-ns
```

4. Vérifiez que les autorisations sont correctes.

   a. Tentative de récupération des informations d'objet AppVault pour tous les espaces de noms :

   ```
   kubectl get appvaults -n trident-protect
   --as=system:serviceaccount:engineering-ns:eng-user
   ```

   Vous devez voir les résultats similaires à ce qui suit :

```
Error from server (Forbidden): appvaults.protect.trident.netapp.io is
forbidden: User "system:serviceaccount:engineering-ns:eng-user"
cannot list resource "appvaults" in API group
"protect.trident.netapp.io" in the namespace "trident-protect"
```

b. Testez pour voir si l'utilisateur peut obtenir les informations AppVault qu'il a maintenant l'autorisation d'accéder :

```
kubectl auth can-i --as=system:serviceaccount:engineering-ns:eng-user
get appvaults.protect.trident.netapp.io/appvault-for-eng-user-only -n
trident-protect
```

Vous devez voir les résultats similaires à ce qui suit :

```
yes
```

Résultat

Les utilisateurs auxquels vous avez accordé des autorisations AppVault doivent pouvoir utiliser des objets AppVault autorisés pour les opérations de gestion des données applicatives et ne doivent pas pouvoir accéder à des ressources en dehors des espaces de noms attribués ou créer de nouvelles ressources auxquelles ils n'ont pas accès.

Générer un bundle de support

Trident Protect permet aux administrateurs de générer des bundles qui incluent des informations utiles pour le support NetApp, notamment des journaux, des metrics et des informations de topologie sur les clusters et les applications à gérer. Si vous êtes connecté à Internet, vous pouvez télécharger des offres groupées de support sur le site de support NetApp (NSS) à l'aide d'un fichier de ressources personnalisées (CR).

Créez un bundle de support à l'aide d'une CR

1. Créez le fichier de ressource personnalisée (CR) et nommez-le (par exemple, `trident-protect-support-bundle.yaml`).
2. Configurez les attributs suivants :
   - **metadata.name**: (*required*) le nom de cette ressource personnalisée; choisissez un nom unique et sensible pour votre environnement.
   - **Spec.triggerType**: (*required*) détermine si le bundle de support est généré immédiatement ou planifié. La génération planifiée du bundle a lieu à 12 h UTC. Valeurs possibles :
     - Planifié
     - Manuel
   - **Spec.uploadEnabled**: (*Optional*) détermine si le bundle de support doit être téléchargé sur le site de support NetApp après sa génération. Si ce n'est pas le cas, la valeur par défaut est `false` . Valeurs possibles :
     - vrai
     - false (valeur par défaut)
   - **Spec.dataWindowStart**: (*Optional*) chaîne de date au format RFC 3339 qui spécifie la date et l'heure auxquelles la fenêtre des données incluses dans le paquet de support doit commencer. Si ce n'est pas le cas, la valeur par défaut est de 24 heures. La date de fenêtre la plus ancienne que vous pouvez spécifier est il y a 7 jours.

     Exemple YAML :

```
apiVersion: protect.trident.netapp.io/v1
kind: AutoSupportBundle
metadata:
  name: trident-protect-support-bundle
spec:
  triggerType: Manual
  uploadEnabled: true
  dataWindowStart: 2024-05-05T12:30:00Z
```

3. Une fois que vous avez rempli le `astra-support-bundle.yaml` fichier avec les valeurs correctes, appliquez la CR :

```
kubectl apply -f trident-protect-support-bundle.yaml
```

Créez un bundle de support à l'aide de l'interface de ligne de commande

1. Créez le pack de support en remplaçant les valeurs entre parenthèses par les informations de votre environnement. `trigger-type`Détermine si le bundle est créé immédiatement ou si l'heure de création est déterminée par le planning, et peut être `Manual` ou `Scheduled`. Le paramètre par défaut est `Manual`.

   Par exemple :

```
tridentctl protect create autosupportbundle <my_bundle_name>
--trigger-type <trigger_type>
```

Gérez et protégez les applications

Utilisez les objets AppVault pour gérer les compartiments

La ressource personnalisée de compartiment (CR) pour Trident Protect est appelée AppVault. Les objets AppVault sont la représentation déclarative du workflow Kubernetes d'un compartiment de stockage. Une CR AppVault contient les configurations nécessaires à l'utilisation d'un compartiment dans les opérations de protection, telles que les sauvegardes, les snapshots, les opérations de restauration et la réplication SnapMirror. Seuls les administrateurs peuvent créer des AppVault.

Exemples de génération de clés et de définition d'AppVault

Lors de la définition d'une CR AppVault, vous devez inclure des informations d'identification pour accéder aux ressources hébergées par le fournisseur. La façon dont vous générez les clés pour les informations d'identification varie en fonction du fournisseur. Vous trouverez ci-dessous des exemples de génération de clés de ligne de commande pour plusieurs fournisseurs, suivis des exemples de définitions AppVault pour chaque fournisseur.

Google Cloud

Exemple de génération de clés :

```
kubectl create secret generic gcp-creds --from-file=credentials=<mycreds
-file.json> -n trident-protect
```

Les exemples de définition d'AppVault suivants sont fournis sous la forme d'une CR que vous pouvez utiliser et modifier, ou sous la forme d'une commande CLI Trident Protect qui génère la CR d'AppVault pour vous :

Exemple de CR AppVault

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: gcp-trident-protect-src-bucket-b643cc50-0429-4ad5-971f-
ac4a83621922
  namespace: trident-protect
spec:
  providerType: GCP
  providerConfig:
    gcp:
      bucketName: trident-protect-src-bucket
      projectID: project-id
  providerCredentials:
    credentials:
      valueFromSecret:
        key: credentials
        name: gcp-trident-protect-src-bucket-secret
```

Exemple de création de CR AppVault à l'aide de l'interface de ligne de commande Trident Protect

```
tridentctl protect create vault gcp my-new-vault --bucket mybucket
--project my-gcp-project --secret <gcp-creds>/<credentials>
```

Amazon S3

Exemple de génération de clés :

```
kubectl create secret generic -n trident-protect s3 --from
-literal=accessKeyID=<secret-name> --from-literal=secretAccessKey
=<generic-s3-trident-protect-src-bucket-secret>
```

Les exemples de définition d'AppVault suivants sont fournis sous la forme d'une CR que vous pouvez utiliser et modifier, ou sous la forme d'une commande CLI Trident Protect qui génère la CR d'AppVault pour vous :

Exemple de CR AppVault

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: amazon-s3-trident-protect-src-bucket-b643cc50-0429-4ad5-971f-
ac4a83621922
  namespace: trident-protect
spec:
  providerType: AWS
  providerConfig:
    s3:
      bucketName: trident-protect-src-bucket
      endpoint: s3.example.com
  providerCredentials:
    accessKeyID:
      valueFromSecret:
        key: accessKeyID
        name: s3
    secretAccessKey:
      valueFromSecret:
        key: secretAccessKey
        name: s3
```

Exemple de création d'AppVault avec CLI

```
tridentctl protect create vault GenericS3 s3vault --bucket <bucket-
name> --secret  <secret-name>  --endpoint <s3-endpoint>
```

Microsoft Azure

Exemple de génération de clés :

```
kubectl create secret generic <secret-name> --from-literal=accountKey
=<secret-name> -n trident-protect
```

Les exemples de définition d'AppVault suivants sont fournis sous la forme d'une CR que vous pouvez utiliser et modifier, ou sous la forme d'une commande CLI Trident Protect qui génère la CR d'AppVault pour vous :

Exemple de CR AppVault

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: azure-trident-protect-src-bucket-b643cc50-0429-4ad5-971f-
ac4a83621922
  namespace: trident-protect
spec:
  providerType: Azure
  providerConfig:
    azure:
      accountName: account-name
      bucketName: trident-protect-src-bucket
  providerCredentials:
    accountKey:
      valueFromSecret:
        key: accountKey
        name: azure-trident-protect-src-bucket-secret
```

Exemple de création d'AppVault avec CLI

```
tridentctl protect create vault Azure <vault-name> --account <account-
name> --bucket <bucket-name> --secret <secret-name>
```

Valeurs prises en charge pour providerType et providerConfig

Les `providerType` clés et `providerConfig` d'une CR AppVault requièrent des valeurs spécifiques. Le tableau suivant répertorie les valeurs prises en charge pour la `providerType` clé et la clé associée `providerConfig` que vous devez utiliser avec chaque `providerType` valeur.

Valeur prise en charge `providerType`	**Clé associée** `providerConfig`
AWS	s3
Azure	azure
GCP	gcp
GenericS3	s3
OntapS3	s3
StorageGridS3	s3

Utilisez le navigateur AppVault pour afficher les informations AppVault

Vous pouvez utiliser le plug-in Trident Protect CLI pour afficher des informations sur les objets AppVault qui ont été créés sur le cluster.

Étapes

1. Afficher le contenu d'un objet AppVault :

```
tridentctl protect get appvaultcontent gcp-vault --show-resources all
```

Exemple de sortie :

```
+-------------+-------+----------+----------------------------
+--------------------------+
|   CLUSTER   |  APP  |   TYPE   |            NAME             |
TIMESTAMP         |
+-------------+-------+----------+----------------------------
+--------------------------+
|             | mysql | snapshot | mysnap                      | 2024-
08-09 21:02:11 (UTC) |
| production1 | mysql | snapshot | hourly-e7db6-20240815180300 | 2024-
08-15 18:03:06 (UTC) |
| production1 | mysql | snapshot | hourly-e7db6-20240815190300 | 2024-
08-15 19:03:06 (UTC) |
| production1 | mysql | snapshot | hourly-e7db6-20240815200300 | 2024-
08-15 20:03:06 (UTC) |
| production1 | mysql | backup   | hourly-e7db6-20240815180300 | 2024-
08-15 18:04:25 (UTC) |
| production1 | mysql | backup   | hourly-e7db6-20240815190300 | 2024-
08-15 19:03:30 (UTC) |
| production1 | mysql | backup   | hourly-e7db6-20240815200300 | 2024-
08-15 20:04:21 (UTC) |
| production1 | mysql | backup   | mybackup5                   | 2024-
08-09 22:25:13 (UTC) |
|             | mysql | backup   | mybackup                    | 2024-
08-09 21:02:52 (UTC) |
+-------------+-------+----------+----------------------------
+--------------------------+
```

2. Si vous le souhaitez, utilisez l'indicateur pour afficher le chemin d'accès à l'application pour chaque ressource `--show-paths` .

   Le nom de cluster figurant dans la première colonne du tableau n'est disponible que si un nom de cluster a été spécifié dans l'installation de Trident Protect Helm. Par exemple : `--set clusterName=production1`.

Supprimer un AppVault

Vous pouvez supprimer un objet AppVault à tout moment.

Ne supprimez pas la `finalizers` clé dans la CR AppVault avant de supprimer l'objet AppVault. Dans ce cas, des données résiduelles dans le compartiment AppVault et des ressources orphelines dans le cluster.

Avant de commencer

Assurez-vous d'avoir supprimé tous les snapshots et les sauvegardes stockés dans le compartiment associé.

Supprimez un AppVault à l'aide de l'interface de ligne de commande Kubernetes

1. Supprimez l'objet AppVault, en le remplaçant `appvault_name` par le nom de l'objet AppVault à supprimer :

```
kubectl delete appvault <appvault_name> -n trident-protect
```

Supprimez un AppVault à l'aide de l'interface de ligne de commande Trident

1. Supprimez l'objet AppVault, en le remplaçant `appvault_name` par le nom de l'objet AppVault à supprimer :

```
tridentctl protect delete appvault <appvault_name> -n trident-protect
```

Définissez une application pour la gestion

Vous pouvez définir une application que vous souhaitez gérer avec Trident Protect en créant une application CR et une application CR AppVault associée.

Créez une CR AppVault

Vous devez créer une CR AppVault qui sera utilisée lors des opérations de protection des données sur l'application et la CR AppVault doit résider sur le cluster sur lequel Trident Protect est installé. La CR AppVault est spécifique à votre environnement ; pour obtenir des exemples de CRS AppVault, reportez-vous à la section "Ressources personnalisées AppVault."

Créer une demande de modification d'application

Vous devez créer une demande de modification pour chaque application que vous souhaitez gérer avec Trident Protect. Vous pouvez ajouter une application à des fins de gestion en créant manuellement une application CR ou en utilisant l'interface de ligne de commande Trident Protect pour créer la CR.

Ajouter une application à l'aide d'une demande de modification

1. Créez le fichier CR de l'application de destination :

   a. Créez le fichier de ressource personnalisée (CR) et nommez-le (par exemple, `maria-app.yaml`).

   b. Configurez les attributs suivants :

      - **metadata.name**: (*required*) le nom de la ressource personnalisée de l'application. Notez le nom que vous choisissez car les autres fichiers CR nécessaires aux opérations de protection font référence à cette valeur.
      - **spec.includedNamespaces**: (*required*) utilisez des étiquettes d'espace de noms ou un nom d'espace de noms pour spécifier les espaces de noms dans lesquels les ressources d'application existent. L'espace de noms de l'application doit faire partie de cette liste.

        Exemple YAML :

```yaml
apiVersion: protect.trident.netapp.io/v1
kind: Application
metadata:
  name: maria
  namespace: my-app-namespace
spec:
  includedNamespaces:
    labelSelector: {}
    namespace: my-app-namespace
```

Ajoutez une application à l'aide de l'interface de ligne de commande

1. Créez et appliquez la définition de l'application, en remplaçant les valeurs entre parenthèses par les informations de votre environnement. Vous pouvez inclure des espaces de noms et des ressources dans la définition d'application à l'aide de listes séparées par des virgules avec les arguments indiqués dans l'exemple suivant :

```
tridentctl protect create application <my_new_app_cr_name>
--namespaces <namespaces_to_include> --csr
<cluster_scoped_resources_to_include>
```

Protégez les applications

Protégez toutes les applications en effectuant des copies Snapshot et des sauvegardes à l'aide d'une stratégie de protection automatisée ou ad hoc.

Créer un snapshot à la demande

Vous pouvez créer un snapshot à la demande à tout moment.

Créer un instantané à l'aide d'une CR

1. Créez le fichier de ressource personnalisée (CR) et nommez-le `trident-protect-snapshot-cr.yaml`.
2. Dans le fichier que vous avez créé, configurez les attributs suivants :
   - **metadata.name**: (*required*) le nom de cette ressource personnalisée; choisissez un nom unique et sensible pour votre environnement.
   - **Spec.applicationRef** : nom Kubernetes de l'application à snapshot.
   - **Spec.appVaultRef**: (*required*) Nom de l'AppVault où le contenu de l'instantané (métadonnées) doit être stocké.
   - **Spec.reclaimPolicy**: (*Optional*) définit ce qui arrive à l'AppArchive d'un snapshot lorsque le snapshot CR est supprimé. Cela signifie que même si la valeurest définie sur `Retain`, l'instantané sera supprimé. Options valides :
     - `Retain` (par défaut)
     - `Delete`

```
apiVersion: protect.trident.netapp.io/v1
kind: Snapshot
metadata:
  namespace: my-app-namespace
  name: my-cr-name
spec:
  applicationRef: my-application
  appVaultRef: appvault-name
  reclaimPolicy: Delete
```

3. Une fois que vous avez rempli le `trident-protect-snapshot-cr.yaml` fichier avec les valeurs correctes, appliquez la CR :

```
kubectl apply -f trident-protect-snapshot-cr.yaml
```

Créer un snapshot à l'aide de l'interface de ligne de commandes

1. Créez l'instantané, en remplaçant les valeurs entre parenthèses par les informations de votre environnement. Par exemple :

```
tridentctl protect create snapshot <my_snapshot_name> --appvault
<my_appvault_name> --app <name_of_app_to_snapshot>
```

Créez une sauvegarde à la demande

Vous pouvez sauvegarder une application à tout moment.

Créez une sauvegarde à l'aide d'une CR

1. Créez le fichier de ressource personnalisée (CR) et nommez-le `trident-protect-backup-cr.yaml`.
2. Dans le fichier que vous avez créé, configurez les attributs suivants :
   - **metadata.name**: (*required*) le nom de cette ressource personnalisée; choisissez un nom unique et sensible pour votre environnement.
   - **Spec.applicationRef**: (*required*) Nom Kubernetes de l'application à sauvegarder.
   - **Spec.appVaultRef**: (*required*) Nom de l'AppVault où le contenu de la sauvegarde doit être stocké.
   - **Spec.datamover**: (*Optional*) chaîne indiquant l'outil de sauvegarde à utiliser pour l'opération de sauvegarde. Valeurs possibles (sensibles à la casse) :
     - `Restic`
     - `Kopia` (par défaut)
   - **Spec.reclaimPolicy**: (*Optional*) définit ce qui arrive à une sauvegarde lorsqu'elle est libérée de sa réclamation. Valeurs possibles :
     - `Delete`
     - `Retain` (par défaut)
   - **Spec.snapshotRef**: (*Optional*): Nom du snapshot à utiliser comme source de la sauvegarde. Si ce n'est pas le cas, un instantané temporaire sera créé et sauvegardé.

   ```
   apiVersion: protect.trident.netapp.io/v1
   kind: Backup
   metadata:
     namespace: my-app-namespace
     name: my-cr-name
   spec:
     applicationRef: my-application
     appVaultRef: appvault-name
     dataMover: Kopia
   ```

3. Une fois que vous avez rempli le `trident-protect-backup-cr.yaml` fichier avec les valeurs correctes, appliquez la CR :

   ```
   kubectl apply -f trident-protect-backup-cr.yaml
   ```

Créez une sauvegarde à l'aide de l'interface de ligne de commande

1. Créez la sauvegarde en remplaçant les valeurs entre parenthèses par les informations de votre environnement. Par exemple :

```
tridentctl protect create backup <my_backup_name> --appvault <my-
vault-name> --app <name_of_app_to_back_up>
```

Créez un calendrier de protection des données

Une règle de protection protège une application en créant des snapshots, des sauvegardes ou les deux à un calendrier défini. Vous pouvez choisir de créer des snapshots et des sauvegardes toutes les heures, tous les jours, toutes les semaines et tous les mois, et vous pouvez spécifier le nombre de copies à conserver.

Créer un programme à l'aide d'une demande de modification

1. Créez le fichier de ressource personnalisée (CR) et nommez-le `trident-protect-schedule-cr.yaml`.
2. Dans le fichier que vous avez créé, configurez les attributs suivants :
   - **metadata.name**: (*required*) le nom de cette ressource personnalisée; choisissez un nom unique et sensible pour votre environnement.
   - **Spec.datamover**: (*Optional*) chaîne indiquant l'outil de sauvegarde à utiliser pour l'opération de sauvegarde. Valeurs possibles (sensibles à la casse) :
     - `Restic`
     - `Kopia` (par défaut)
   - **Spec.applicationRef** : nom Kubernetes de l'application à sauvegarder.
   - **Spec.appVaultRef**: (*required*) Nom de l'AppVault où le contenu de la sauvegarde doit être stocké.
   - **Spec.backupRetention** : le nombre de sauvegardes à conserver. Zéro indique qu'aucune sauvegarde ne doit être créée.
   - **Spec.snapshotRetention** : le nombre d'instantanés à conserver. Zéro indique qu'aucun snapshot ne doit être créé.
   - **spec.granularity**: la fréquence à laquelle le programme doit s'exécuter. Valeurs possibles, ainsi que les champs associés obligatoires :
     - `hourly` (nécessite que vous spécifiiez `spec.minute`)
     - `daily` (nécessite que vous spécifiiez `spec.minute` et `spec.hour`)
     - `weekly` (nécessite que vous spécifiiez `spec.minute, spec.hour`, et `spec.dayOfWeek`)
     - `monthly` (nécessite que vous spécifiiez `spec.minute, spec.hour`, et `spec.dayOfMonth`)
   - **Spec.dayOfMonth**: (*Optional*) le jour du mois (1 - 31) que le programme doit s'exécuter. Ce champ est obligatoire si la granularité est définie sur `monthly`.
   - **Spec.dayOfWeek**: (*Optional*) le jour de la semaine (0 - 7) que le programme doit s'exécuter. Les valeurs 0 ou 7 indiquent dimanche. Ce champ est obligatoire si la granularité est définie sur `weekly`.
   - **Spec.hour**: (*Optional*) heure du jour (0 - 23) que le programme doit exécuter. Ce champ est obligatoire si la granularité est définie sur `daily`, `weekly`ou `monthly`.
   - **Spec.minute**: (*Optional*) la minute de l'heure (0 - 59) que le programme doit exécuter. Ce champ est obligatoire si la granularité est définie sur `hourly`, `daily`, `weekly`ou `monthly`.

```
apiVersion: protect.trident.netapp.io/v1
kind: Schedule
metadata:
  namespace: my-app-namespace
  name: my-cr-name
spec:
  dataMover: Kopia
  applicationRef: my-application
  appVaultRef: appvault-name
  backupRetention: "15"
  snapshotRetention: "15"
  granularity: <monthly>
  dayOfMonth: "1"
  dayOfWeek: "0"
  hour: "0"
  minute: "0"
```

3. Une fois que vous avez rempli le `trident-protect-schedule-cr.yaml` fichier avec les valeurs correctes, appliquez la CR :

```
kubectl apply -f trident-protect-schedule-cr.yaml
```

Créez un planning à l'aide de l'interface de ligne de commandes

1. Créez le planning de protection en remplaçant les valeurs entre parenthèses par les informations de votre environnement. Par exemple :

> Vous pouvez utiliser `tridentctl protect create schedule --help` pour afficher les informations d'aide détaillées de cette commande.

```
tridentctl protect create schedule <my_schedule_name> --appvault
<my_appvault_name> --app <name_of_app_to_snapshot> --backup
-retention <how_many_backups_to_retain> --data-mover
<kopia_or_restic> --day-of-month <day_of_month_to_run_schedule>
--day-of-week <day_of_month_to_run_schedule> --granularity
<frequency_to_run> --hour <hour_of_day_to_run> --minute
<minute_of_hour_to_run> --recurrence-rule <recurrence> --snapshot
-retention <how_many_snapshots_to_retain>
```

Supprime un snapshot

Supprimez les snapshots programmés ou à la demande dont vous n'avez plus besoin.

Étapes

1. Supprimer l'instantané CR associé à l'instantané :

```
kubectl delete snapshot <snapshot_name> -n my-app-namespace
```

Supprimer une sauvegarde

Supprimez les sauvegardes planifiées ou à la demande qui ne vous sont plus nécessaires.

Étapes

1. Supprimez la CR de sauvegarde associée à la sauvegarde :

```
kubectl delete backup <backup_name> -n my-app-namespace
```

Vérifier l'état d'une opération de sauvegarde

Vous pouvez utiliser la ligne de commande pour vérifier l'état d'une opération de sauvegarde en cours, terminée ou ayant échoué.

Étapes

1. Utiliser la commande suivante pour récupérer le statut de l'opération de sauvegarde en remplaçant les valeurs entre crochets par des informations de votre environnement :

```
kubectl get backup -n <namespace_name> <my_backup_cr_name> -o jsonpath
='{.status}'
```

Activez la sauvegarde et la restauration pour les opérations Azure-NetApp-Files (ANF)

Si vous avez installé Trident Protect, vous pouvez activer la fonctionnalité de sauvegarde et de restauration compactes pour les systèmes back-end qui utilisent la classe de stockage Azure-NetApp-Files et qui ont été créés avant Trident 24.06. Cette fonctionnalité fonctionne avec les volumes NFSv4 et ne consomme pas d'espace supplémentaire dans le pool de capacité.

Avant de commencer

Vérifiez les points suivants :

- Vous avez installé Trident Protect.
- Vous avez défini une application dans Trident Protect. Cette application aura une fonctionnalité de protection limitée jusqu'à ce que vous ayez terminé cette procédure.
- Vous avez `azure-netapp-files` sélectionné comme classe de stockage par défaut pour votre système back-end de stockage.

Développez pour les étapes de configuration

1. Si le volume ANF a été créé avant la mise à niveau vers Trident 24.10, procédez comme suit dans Trident :

   a. Activez le répertoire Snapshot pour chaque volume persistant basé sur Azure-NetApp-Files et associé à l'application :

   ```
   tridentctl update volume <pv name> --snapshot-dir=true -n trident
   ```

   b. Vérifiez que le répertoire de snapshot a été activé pour chaque PV associé :

   ```
   tridentctl get volume <pv name> -n trident -o yaml | grep
   snapshotDir
   ```

   Réponse :

   ```
   snapshotDirectory: "true"
   ```

   +
   Lorsque le répertoire de snapshots n'est pas activé, Trident Protect choisit la fonctionnalité de sauvegarde standard, qui consomme temporairement de l'espace dans le pool de capacité pendant le processus de sauvegarde. Dans ce cas, assurez-vous que l'espace disponible dans le pool de capacité est suffisant pour créer un volume temporaire de la taille du volume en cours de sauvegarde.

Résultat

L'application est prête pour la sauvegarde et la restauration à l'aide de Trident Protect. Chaque demande de volume persistant est également disponible pour être utilisée par d'autres applications à des fins de sauvegarde et de restauration.

Restauration des applications

Vous pouvez utiliser Trident Protect pour restaurer votre application à partir d'une copie Snapshot ou d'une sauvegarde. La restauration d'un snapshot existant est plus rapide lors de la restauration d'une application sur le même cluster.

> Lorsque vous restaurez une application, tous les crochets d'exécution configurés pour l'application sont restaurés avec l'application. Si un hook d'exécution post-restauration est présent, il s'exécute automatiquement dans le cadre de l'opération de restauration.

Restauration d'une sauvegarde vers un autre espace de noms

Lorsque vous restaurez une sauvegarde dans un espace de noms différent à l'aide d'une sauvegarde CR BackupRestore, Trident Protect restaure l'application dans un nouvel espace de noms, mais l'application restaurée n'est pas automatiquement protégée par Trident Protect. Pour protéger l'application restaurée, vous devez créer une application CR pour l'application restaurée afin qu'elle soit protégée par Trident Protect.

La restauration d'une sauvegarde dans un espace de noms différent avec des ressources existantes ne modifie aucune ressource qui partage des noms avec ceux de la sauvegarde. Pour restaurer toutes les ressources de la sauvegarde, supprimez et recréez l'espace de noms cible ou restaurez la sauvegarde dans un nouvel espace de noms.

Utiliser une CR

1. Créez le fichier de ressource personnalisée (CR) et nommez-le `trident-protect-backup-restore-cr.yaml`.
2. Dans le fichier que vous avez créé, configurez les attributs suivants :
   - **metadata.name**: (*required*) le nom de cette ressource personnalisée; choisissez un nom unique et sensible pour votre environnement.
   - **Spec.appArchivePath** : chemin d'accès dans AppVault où sont stockés le contenu de la sauvegarde. Vous pouvez utiliser la commande suivante pour trouver ce chemin :

     ```
     kubectl get backups <BACKUP_NAME> -n my-app-namespace -o
     jsonpath='{.status.appArchivePath}'
     ```

   - **Spec.appVaultRef**: (*required*) Nom de l'AppVault où sont stockés le contenu de la sauvegarde.
   - **spec.namespaceMapping**: mappage de l'espace de noms source de l'opération de restauration sur l'espace de noms de destination. Remplacez `my-source-namespace` et `my-destination-namespace` par des informations provenant de votre environnement.
   - **Spec.storageClassMapping** : mappage de la classe de stockage source de l'opération de restauration à la classe de stockage de destination. Remplacez `destinationStorageClass` et `sourceStorageClass` par des informations provenant de votre environnement.

     ```
     apiVersion: protect.trident.netapp.io/v1o
     kind: BackupRestore
     metadata:
       name: my-cr-name
       namespace: my-app-namespace
     spec:
       appArchivePath: my-backup-path
       appVaultRef: appvault-name
       namespaceMapping: [{"source": "my-source-namespace",
     "destination": "my-destination-namespace"}]
       storageClassMapping:
         destination: "${destinationStorageClass}"
         source: "${sourceStorageClass}"
     ```

3. (*Facultatif*) si vous devez sélectionner uniquement certaines ressources de l'application à restaurer, ajoutez un filtrage qui inclut ou exclut les ressources marquées avec des étiquettes particulières :
   - **ResourceFilter.resourceSelectionCriteria**: (Requis pour le filtrage) utilisez `include or exclude` pour inclure ou exclure une ressource définie dans resourceMatchers. Ajoutez les paramètres resourceMatchers suivants pour définir les ressources à inclure ou à exclure :
     - **ResourceFilter.resourceMatchers** : tableau des objets resourceMatcher.
       - **ResourceMatchers[].group**: (*Optional*) Groupe de la ressource à filtrer.
       - **ResourceMatchers[].kind**: (*Optional*) Type de la ressource à filtrer.

- **ResourceMatchers[].version**: (*Optional*) version de la ressource à filtrer.
- **ResourceMatchers[].names**: (*Optional*) noms dans le champ Kubernetes metadata.name de la ressource à filtrer.
- **ResourceMatchers[].namespaces**: (*Optional*) Namespaces dans le champ Kubernetes metadata.name de la ressource à filtrer.
- **ResourceMatchers[].labelSelectors**: (*Optional*) chaîne de sélecteur de libellé dans le champ Kubernetes metadata.name de la ressource, comme défini dans le "Documentation Kubernetes". Par exemple : `"trident.netapp.io/os=linux"`.

  Par exemple :

```
spec:
  resourceFilter:
    resourceSelectionCriteria: "include"
    resourceMatchers:
      group: my-resource-group
      kind: my-resource-kind
      version: my-resource-version
      names: ["my-resource-names"]
      namespaces: ["my-resource-namespaces"]
      labelSelectors: ["trident.netapp.io/os=linux"]
```

4. Une fois que vous avez rempli le `trident-protect-backup-restore-cr.yaml` fichier avec les valeurs correctes, appliquez la CR :

```
kubectl apply -f trident-protect-backup-restore-cr.yaml
```

Utiliser l'interface de ligne de commande

1. Restaurez la sauvegarde dans un espace de noms différent, en remplaçant les valeurs entre parenthèses par les informations de votre environnement. L' `namespace-mapping`argument utilise des espaces de noms séparés par deux-points pour mapper les espaces de noms source aux espaces de noms de destination corrects dans le format `source1:dest1,source2:dest2`. Par exemple :

```
tridentctl protect create backuprestore <my_restore_name> --backup
<backup_namespace>/<backup_to_restore> --namespace-mapping
<source_to_destination_namespace_mapping>
```

Restaurer à partir d'une sauvegarde vers l'espace de noms d'origine

Vous pouvez à tout moment restaurer une sauvegarde dans l'espace de noms d'origine.

Utiliser une CR

1. Créez le fichier de ressource personnalisée (CR) et nommez-le `trident-protect-backup-ipr-cr.yaml`.
2. Dans le fichier que vous avez créé, configurez les attributs suivants :
   - **metadata.name**: (*required*) le nom de cette ressource personnalisée; choisissez un nom unique et sensible pour votre environnement.
   - **Spec.appArchivePath** : chemin d'accès dans AppVault où sont stockés le contenu de la sauvegarde. Vous pouvez utiliser la commande suivante pour trouver ce chemin :

     ```
     kubectl get backups <BACKUP_NAME> -n my-app-namespace -o
     jsonpath='{.status.appArchivePath}'
     ```

   - **Spec.appVaultRef**: (*required*) Nom de l'AppVault où sont stockés le contenu de la sauvegarde.

     Par exemple :

```
apiVersion: protect.trident.netapp.io/v1
kind: BackupInplaceRestore
metadata:
  name: my-cr-name
  namespace: my-app-namespace
spec:
  appArchivePath: my-backup-path
  appVaultRef: appvault-name
```

3. (*Facultatif*) si vous devez sélectionner uniquement certaines ressources de l'application à restaurer, ajoutez un filtrage qui inclut ou exclut les ressources marquées avec des étiquettes particulières :
   - **ResourceFilter.resourceSelectionCriteria**: (Requis pour le filtrage) utilisez `include or exclude` pour inclure ou exclure une ressource définie dans resourceMatchers. Ajoutez les paramètres resourceMatchers suivants pour définir les ressources à inclure ou à exclure :
     - **ResourceFilter.resourceMatchers** : tableau des objets resourceMatcher.
       - **ResourceMatchers[].group**: (*Optional*) Groupe de la ressource à filtrer.
       - **ResourceMatchers[].kind**: (*Optional*) Type de la ressource à filtrer.
       - **ResourceMatchers[].version**: (*Optional*) version de la ressource à filtrer.
       - **ResourceMatchers[].names**: (*Optional*) noms dans le champ Kubernetes metadata.name de la ressource à filtrer.
       - **ResourceMatchers[].namespaces**: (*Optional*) Namespaces dans le champ Kubernetes metadata.name de la ressource à filtrer.
       - **ResourceMatchers[].labelSelectors**: (*Optional*) chaîne de sélecteur de libellé dans le champ Kubernetes metadata.name de la ressource, comme défini dans le "Documentation Kubernetes". Par exemple : `"trident.netapp.io/os=linux"`.

Par exemple :

```
spec:
  resourceFilter:
    resourceSelectionCriteria: "include"
    resourceMatchers:
      group: my-resource-group
      kind: my-resource-kind
      version: my-resource-version
      names: ["my-resource-names"]
      namespaces: ["my-resource-namespaces"]
      labelSelectors: ["trident.netapp.io/os=linux"]
```

4. Une fois que vous avez rempli le `trident-protect-backup-ipr-cr.yaml` fichier avec les valeurs correctes, appliquez la CR :

```
kubectl apply -f trident-protect-backup-ipr-cr.yaml
```

Utiliser l'interface de ligne de commande

1. Restaurez la sauvegarde dans l'espace de noms d'origine en remplaçant les valeurs entre parenthèses par les informations de votre environnement. L'`backup`argument utilise un nom d'espace de noms et un nom de sauvegarde au format `<namespace>/<name>`. Par exemple :

```
tridentctl protect create backupinplacerestore <my_restore_name>
--backup <namespace/backup_to_restore>
```

Restauration d'un snapshot vers un autre espace de noms

Vous pouvez restaurer les données d'un instantané à l'aide d'un fichier de ressource personnalisée (CR) dans un espace de noms différent ou dans l'espace de noms source d'origine. Lorsque vous restaurez un snapshot sur un autre espace de noms à l'aide d'un CR SnapshotRestore, Trident Protect restaure l'application dans un nouvel espace de noms, mais l'application restaurée n'est pas automatiquement protégée par Trident Protect. Pour protéger l'application restaurée, vous devez créer une application CR pour l'application restaurée afin qu'elle soit protégée par Trident Protect.

Utiliser une CR

1. Créez le fichier de ressource personnalisée (CR) et nommez-le `trident-protect-snapshot-restore-cr.yaml`.
2. Dans le fichier que vous avez créé, configurez les attributs suivants :
   - **metadata.name**: (*required*) le nom de cette ressource personnalisée; choisissez un nom unique et sensible pour votre environnement.
   - **Spec.appVaultRef**: (*required*) le nom du AppVault dans lequel le contenu de l'instantané est stocké.
   - **Spec.appArchivePath** : chemin d'accès dans AppVault où sont stockés le contenu de l'instantané. Vous pouvez utiliser la commande suivante pour trouver ce chemin :

     ```
     kubectl get snapshots <SNAPHOT_NAME> -n my-app-namespace -o jsonpath='{.status.appArchivePath}'
     ```

   - **spec.namespaceMapping**: mappage de l'espace de noms source de l'opération de restauration sur l'espace de noms de destination. Remplacez `my-source-namespace` et `my-destination-namespace` par des informations provenant de votre environnement.
   - **Spec.storageClassMapping** : mappage de la classe de stockage source de l'opération de restauration à la classe de stockage de destination. Remplacez `destinationStorageClass` et `sourceStorageClass` par des informations provenant de votre environnement.

     ```
     apiVersion: protect.trident.netapp.io/v1
     kind: SnapshotRestore
     metadata:
       name: my-cr-name
       namespace: my-app-namespace
     spec:
       appVaultRef: appvault-name
       appArchivePath: my-snapshot-path
       namespaceMapping: [{"source": "my-source-namespace",
     "destination": "my-destination-namespace"}]
       storageClassMapping:
         destination: "${destinationStorageClass}"
         source: "${sourceStorageClass}"
     ```

3. (*Facultatif*) si vous devez sélectionner uniquement certaines ressources de l'application à restaurer, ajoutez un filtrage qui inclut ou exclut les ressources marquées avec des étiquettes particulières :
   - **ResourceFilter.resourceSelectionCriteria**: (Requis pour le filtrage) utilisez `include or exclude` pour inclure ou exclure une ressource définie dans resourceMatchers. Ajoutez les paramètres resourceMatchers suivants pour définir les ressources à inclure ou à exclure :
     - **ResourceFilter.resourceMatchers** : tableau des objets resourceMatcher.
       - **ResourceMatchers[].group**: (*Optional*) Groupe de la ressource à filtrer.

- **ResourceMatchers[].kind**: (*Optional*) Type de la ressource à filtrer.
- **ResourceMatchers[].version**: (*Optional*) version de la ressource à filtrer.
- **ResourceMatchers[].names**: (*Optional*) noms dans le champ Kubernetes metadata.name de la ressource à filtrer.
- **ResourceMatchers[].namespaces**: (*Optional*) Namespaces dans le champ Kubernetes metadata.name de la ressource à filtrer.
- **ResourceMatchers[].labelSelectors**: (*Optional*) chaîne de sélecteur de libellé dans le champ Kubernetes metadata.name de la ressource, comme défini dans le "Documentation Kubernetes". Par exemple : `"trident.netapp.io/os=linux"`.

  Par exemple :

```
spec:
  resourceFilter:
    resourceSelectionCriteria: "include"
    resourceMatchers:
      group: my-resource-group
      kind: my-resource-kind
      version: my-resource-version
      names: ["my-resource-names"]
      namespaces: ["my-resource-namespaces"]
      labelSelectors: ["trident.netapp.io/os=linux"]
```

4. Une fois que vous avez rempli le `trident-protect-snapshot-restore-cr.yaml` fichier avec les valeurs correctes, appliquez la CR :

```
kubectl apply -f trident-protect-snapshot-restore-cr.yaml
```

Utiliser l'interface de ligne de commande

1. Restaurez l'instantané dans un autre espace de noms, en remplaçant les valeurs entre parenthèses par les informations de votre environnement.

   - L'`snapshot`argument utilise un nom d'espace de noms et un nom d'instantané au format `<namespace>/<name>.
   - L'`namespace-mapping`argument utilise des espaces de noms séparés par deux-points pour mapper les espaces de noms source aux espaces de noms de destination corrects dans le format `source1:dest1,source2:dest2.

     Par exemple :

```
tridentctl protect create snapshotrestore <my_restore_name>
--snapshot <namespace/snapshot_to_restore> --namespace-mapping
<source_to_destination_namespace_mapping>
```

Restaurer à partir d’un snapshot vers l’espace de noms d’origine

Vous pouvez à tout moment restaurer un snapshot dans l’espace de noms d’origine.

Utiliser une CR

1. Créez le fichier de ressource personnalisée (CR) et nommez-le `trident-protect-snapshot-ipr-cr.yaml`.
2. Dans le fichier que vous avez créé, configurez les attributs suivants :
   - **metadata.name**: (*required*) le nom de cette ressource personnalisée; choisissez un nom unique et sensible pour votre environnement.
   - **Spec.appVaultRef**: (*required*) le nom du AppVault dans lequel le contenu de l'instantané est stocké.
   - **Spec.appArchivePath** : chemin d'accès dans AppVault où sont stockés le contenu de l'instantané. Vous pouvez utiliser la commande suivante pour trouver ce chemin :

     ```
     kubectl get snapshots <SNAPSHOT_NAME> -n my-app-namespace -o jsonpath='{.status.appArchivePath}'
     ```

     ```
     apiVersion: protect.trident.netapp.io/v1
     kind: SnapshotInplaceRestore
     metadata:
       name: my-cr-name
       namespace: my-app-namespace
     spec:
       appVaultRef: appvault-name
         appArchivePath: my-snapshot-path
     ```

3. (*Facultatif*) si vous devez sélectionner uniquement certaines ressources de l'application à restaurer, ajoutez un filtrage qui inclut ou exclut les ressources marquées avec des étiquettes particulières :
   - **ResourceFilter.resourceSelectionCriteria**: (Requis pour le filtrage) utilisez `include or exclude` pour inclure ou exclure une ressource définie dans resourceMatchers. Ajoutez les paramètres resourceMatchers suivants pour définir les ressources à inclure ou à exclure :
     - **ResourceFilter.resourceMatchers** : tableau des objets resourceMatcher.
       - **ResourceMatchers[].group**: (*Optional*) Groupe de la ressource à filtrer.
       - **ResourceMatchers[].kind**: (*Optional*) Type de la ressource à filtrer.
       - **ResourceMatchers[].version**: (*Optional*) version de la ressource à filtrer.
       - **ResourceMatchers[].names**: (*Optional*) noms dans le champ Kubernetes metadata.name de la ressource à filtrer.
       - **ResourceMatchers[].namespaces**: (*Optional*) Namespaces dans le champ Kubernetes metadata.name de la ressource à filtrer.
       - **ResourceMatchers[].labelSelectors**: (*Optional*) chaîne de sélecteur de libellé dans le champ Kubernetes metadata.name de la ressource, comme défini dans le "Documentation Kubernetes". Par exemple : `"trident.netapp.io/os=linux"`.

         Par exemple :

```
spec:
  resourceFilter:
    resourceSelectionCriteria: "include"
    resourceMatchers:
      group: my-resource-group
      kind: my-resource-kind
      version: my-resource-version
      names: ["my-resource-names"]
      namespaces: ["my-resource-namespaces"]
      labelSelectors: ["trident.netapp.io/os=linux"]
```

4. Une fois que vous avez rempli le `trident-protect-snapshot-ipr-cr.yaml` fichier avec les valeurs correctes, appliquez la CR :

```
kubectl apply -f trident-protect-snapshot-ipr-cr.yaml
```

Utiliser l'interface de ligne de commande

1. Restaurez l'instantané dans l'espace de noms d'origine en remplaçant les valeurs entre parenthèses par les informations de votre environnement. Par exemple :

```
tridentctl protect create snapshotinplacerestore <my_restore_name>
--snapshot <snapshot_to_restore>
```

Vérifiez l'état d'une opération de restauration

Vous pouvez utiliser la ligne de commande pour vérifier l'état d'une opération de restauration en cours, terminée ou ayant échoué.

Étapes

1. Utilisez la commande suivante pour récupérer le statut de l'opération de restauration en remplaçant les valeurs entre crochets par des informations de votre environnement :

```
kubectl get backuprestore -n <namespace_name> <my_restore_cr_name> -o
jsonpath='{.status}'
```

Répliquez vos applications avec NetApp SnapMirror

Avec Trident Protect, vous pouvez utiliser les fonctionnalités de réplication asynchrone de la technologie NetApp SnapMirror pour répliquer les modifications des données et des applications d'un système back-end à un autre, sur le même cluster ou entre différents clusters.

Configuration d’une relation de réplication

La configuration d’une relation de réplication implique les éléments suivants :

- Choix de la fréquence à laquelle Trident Protect doit créer une copie Snapshot d’application (qui inclut les ressources Kubernetes de l’application ainsi que les snapshots de volume pour chacun des volumes de l’application)
- Choix de la planification de la réplication (inclut les ressources Kubernetes ainsi que les données de volume persistant)
- Définition de la durée de prise de l’instantané

Étapes

1. Créez un AppVault pour l’application source sur le cluster source. Selon votre fournisseur de stockage, modifiez un exemple en fonction de "Ressources personnalisées AppVault"votre environnement :

Créez un AppVault à l'aide d'une CR

a. Créez le fichier de ressource personnalisée (CR) et nommez-le (par exemple, `trident-protect-appvault-primary-source.yaml`).

b. Configurez les attributs suivants :

- **metadata.name**: (*required*) le nom de la ressource personnalisée AppVault. Notez le nom que vous choisissez, car les autres fichiers CR nécessaires pour une relation de réplication font référence à cette valeur.
- **spec.providerConfig**: (*required*) stocke la configuration nécessaire pour accéder à AppVault à l'aide du fournisseur spécifié. Choisissez un nom de bucketName et tout autre détail nécessaire pour votre fournisseur. Notez les valeurs que vous choisissez, car les autres fichiers CR nécessaires à une relation de réplication font référence à ces valeurs. Reportez-vous à la section "Ressources personnalisées AppVault" pour obtenir des exemples de CRS AppVault avec d'autres fournisseurs.
- **spec.providerCredentials**: (*required*) stocke les références à toute information d'identification requise pour accéder à AppVault à l'aide du fournisseur spécifié.
  - **spec.providerCredentials.valueFromSecret**: (*required*) indique que la valeur d'identification doit provenir d'un secret.
    - **Key**: (*required*) la clé valide du secret à sélectionner.
    - **Name**: (*required*) Nom du secret contenant la valeur de ce champ. Doit être dans le même espace de noms.
  - **spec.providerCredentials.secretAccessKey**: (*required*) la clé d'accès utilisée pour accéder au fournisseur. Le **nom** doit correspondre à **spec.providerCredentials.valueFromSecret.name**.
- **spec.providerType**: (*required*) détermine ce qui permet la sauvegarde, par exemple NetApp ONTAP S3, S3 générique, Google Cloud ou Microsoft Azure. Valeurs possibles :
  - aws
  - azure
  - gcp
  - générique-s3
  - ONTAP s3
  - StorageGRID s3

c. Une fois que vous avez rempli le `trident-protect-appvault-primary-source.yaml` fichier avec les valeurs correctes, appliquez la CR :

```
kubectl apply -f trident-protect-appvault-primary-source.yaml -n
trident-protect
```

Créez un AppVault à l'aide de la CLI

a. Créez AppVault, en remplaçant les valeurs entre parenthèses par les informations de votre environnement :

```
tridentctl protect create vault Azure <vault-name> --account
<account-name> --bucket <bucket-name> --secret <secret-name>
```

2. Créez l'application source CR :

Créez l'application source à l'aide d'une demande de modification

a. Créez le fichier de ressource personnalisée (CR) et nommez-le (par exemple, `trident-protect-app-source.yaml`).

b. Configurez les attributs suivants :

- **metadata.name**: (*required*) le nom de la ressource personnalisée de l'application. Notez le nom que vous choisissez, car les autres fichiers CR nécessaires pour une relation de réplication font référence à cette valeur.
- **spec.includedNamespaces**: (*required*) un tableau d'espaces de noms et d'étiquettes associées. Utilisez des noms d'espace de noms et, éventuellement, affinez la portée des espaces de noms avec des étiquettes pour spécifier les ressources qui existent dans les espaces de noms répertoriés ici. L'espace de nom de l'application doit faire partie de ce tableau.

  Exemple YAML :

```
apiVersion: protect.trident.netapp.io/v1
kind: Application
metadata:
  name: maria
  namespace: my-app-namespace
spec:
  includedNamespaces:
    - namespace: maria
      labelSelector: {}
```

c. Une fois que vous avez rempli le `trident-protect-app-source.yaml` fichier avec les valeurs correctes, appliquez la CR :

```
kubectl apply -f trident-protect-app-source.yaml -n my-app-namespace
```

Créez l'application source à l'aide de l'interface de ligne de commande

a. Créez l'application source. Par exemple :

```
tridentctl protect create app maria --namespaces maria -n my-app-namespace
```

3. Vous pouvez également créer un snapshot de l'application source. Ce snapshot est utilisé comme base pour l'application sur le cluster de destination. Si vous ignorez cette étape, vous devez attendre l'exécution du prochain snapshot planifié pour avoir un instantané récent.

Prendre un instantané à l'aide d'une CR

a. Créez un planning de réplication pour l'application source :

i. Créez le fichier de ressource personnalisée (CR) et nommez-le (par exemple, `trident-protect-schedule.yaml`).

ii. Configurez les attributs suivants :

- **metadata.name**: (*required*) le nom de la ressource personnalisée d'horaire.
- **Spec.AppVaultRef**: (*required*) cette valeur doit correspondre au champ metadata.name de l'AppVault pour l'application source.
- **Spec.ApplicationRef**: (*required*) cette valeur doit correspondre au champ metadata.name de l'application source CR.
- **Spec.backupRetention**: (*required*) ce champ est obligatoire et la valeur doit être définie sur 0.
- **Spec.enabled** : doit être défini sur true.
- **spec.granularity**: doit être défini sur `Custom`.
- **Spec.recurrenceRule** : définissez une date de début en heure UTC et un intervalle de récurrence.
- **Spec.snapshotRetention** : doit être défini sur 2.

Exemple YAML :

```
apiVersion: protect.trident.netapp.io/v1
kind: Schedule
metadata:
  name: appmirror-schedule-0e1f88ab-f013-4bce-8ae9-6afed9df59a1
  namespace: my-app-namespace
spec:
  appVaultRef: generic-s3-trident-protect-src-bucket-04b6b4ec-
46a3-420a-b351-45795e1b5e34
  applicationRef: maria
  backupRetention: "0"
  enabled: true
  granularity: custom
  recurrenceRule: |-
    DTSTART:20220101T000200Z
    RRULE:FREQ=MINUTELY;INTERVAL=5
  snapshotRetention: "2"
```

i. Une fois que vous avez rempli le `trident-protect-schedule.yaml` fichier avec les valeurs correctes, appliquez la CR :

```
kubectl apply -f trident-protect-schedule.yaml -n my-app-
namespace
```

Créer un snapshot à l'aide de l'interface de ligne de commande

a. Créez l'instantané, en remplaçant les valeurs entre parenthèses par les informations de votre environnement. Par exemple :

```
tridentctl protect create snapshot <my_snapshot_name> --appvault
<my_appvault_name> --app <name_of_app_to_snapshot>
```

4. Créez une application source AppVault CR sur le cluster de destination qui est identique à la CR AppVault que vous avez appliquée sur le cluster source et nommez-la (par exemple, `trident-protect-appvault-primary-destination.yaml`).
5. Appliquer la CR :

```
kubectl apply -f trident-protect-appvault-primary-destination.yaml -n
my-app-namespace
```

6. Créez un AppVault pour l'application de destination sur le cluster de destination. Selon votre fournisseur de stockage, modifiez un exemple en fonction de "Ressources personnalisées AppVault"votre environnement :

   a. Créez le fichier de ressource personnalisée (CR) et nommez-le (par exemple, `trident-protect-appvault-secondary-destination.yaml`).

   b. Configurez les attributs suivants :

      - **metadata.name**: (*required*) le nom de la ressource personnalisée AppVault. Notez le nom que vous choisissez, car les autres fichiers CR nécessaires pour une relation de réplication font référence à cette valeur.
      - **spec.providerConfig**: (*required*) stocke la configuration nécessaire pour accéder à AppVault à l'aide du fournisseur spécifié. Choisissez un `bucketName` et d'autres détails nécessaires pour votre fournisseur. Notez les valeurs que vous choisissez, car les autres fichiers CR nécessaires à une relation de réplication font référence à ces valeurs. Reportez-vous à la section "Ressources personnalisées AppVault" pour obtenir des exemples de CRS AppVault avec d'autres fournisseurs.
      - **spec.providerCredentials**: (*required*) stocke les références à toute information d'identification requise pour accéder à AppVault à l'aide du fournisseur spécifié.
        - **spec.providerCredentials.valueFromSecret**: (*required*) indique que la valeur d'identification doit provenir d'un secret.
          - **Key**: (*required*) la clé valide du secret à sélectionner.
          - **Name**: (*required*) Nom du secret contenant la valeur de ce champ. Doit être dans le même espace de noms.
        - **spec.providerCredentials.secretAccessKey**: (*required*) la clé d'accès utilisée pour accéder au fournisseur. Le **nom** doit correspondre à

spec.providerCredentials.valueFromSecret.name.

- **spec.providerType**: (*required*) détermine ce qui permet la sauvegarde, par exemple NetApp ONTAP S3, S3 générique, Google Cloud ou Microsoft Azure. Valeurs possibles :
  - aws
  - azure
  - gcp
  - générique-s3
  - ONTAP s3
  - StorageGRID s3

c. Une fois que vous avez rempli le `trident-protect-appvault-secondary-destination.yaml` fichier avec les valeurs correctes, appliquez la CR :

```
kubectl apply -f trident-protect-appvault-secondary-destination.yaml
-n my-app-namespace
```

7. Créez un fichier CR AppMirrorRelationship :

Créez un AppMirrorRelationship à l'aide d'une CR

a. Créez le fichier de ressource personnalisée (CR) et nommez-le (par exemple, `trident-protect-relationship.yaml`).

b. Configurez les attributs suivants :

- **metadata.name:** (obligatoire) le nom de la ressource personnalisée AppMirrorRelationship.
- **spec.destinationAppVaultRef**: (*required*) cette valeur doit correspondre au nom de l'AppVault pour l'application de destination sur le cluster de destination.
- **spec.namespaceMapping**: (*required*) les espaces de noms de destination et de source doivent correspondre à l'espace de noms d'application défini dans la CR de l'application correspondante.
- **Spec.sourceAppVaultRef**: (*required*) cette valeur doit correspondre au nom du AppVault pour l'application source.
- **Spec.sourceApplicationName**: (*required*) cette valeur doit correspondre au nom de l'application source que vous avez définie dans la CR de l'application source.
- **Spec.storageClassName**: (*required*) Choisissez le nom d'une classe de stockage valide sur le cluster. La classe de stockage doit être associée à la classe de stockage utilisée sur le cluster source sur lequel l'application source est déployée.
- **Spec.recurrenceRule** : définissez une date de début en heure UTC et un intervalle de récurrence.

Exemple YAML :

```
apiVersion: protect.trident.netapp.io/v1
kind: AppMirrorRelationship
metadata:
  name: amr-16061e80-1b05-4e80-9d26-d326dc1953d8
  namespace: my-app-namespace
spec:
  desiredState: Established
  destinationAppVaultRef: generic-s3-trident-protect-dst-bucket-
8fe0b902-f369-4317-93d1-ad7f2edc02b5
  namespaceMapping:
    - destination: my-app-namespace
      source: my-app-namespace
  recurrenceRule: |-
    DTSTART:20220101T000200Z
    RRULE:FREQ=MINUTELY;INTERVAL=5
  sourceAppVaultRef: generic-s3-trident-protect-src-bucket-
b643cc50-0429-4ad5-971f-ac4a83621922
  sourceApplicationName: maria
  sourceApplicationUID: 7498d32c-328e-4ddd-9029-122540866aeb
  storageClassName: sc-vsim-2
```

c. Une fois que vous avez rempli le `trident-protect-relationship.yaml` fichier avec les valeurs correctes, appliquez la CR :

```
kubectl apply -f trident-protect-relationship.yaml -n my-app-namespace
```

Créez un AppMirrorRelationship à l'aide de l'interface de ligne de commande

a. Créez et appliquez l'objet AppMirrorRelationship, en remplaçant les valeurs entre parenthèses par les informations de votre environnement. Par exemple :

```
tridentctl protect create appmirrorrelationship
<name_of_appmirorrelationship> --destination-app-vault
<my_vault_name> --recurrence-rule <rule> --source-app
<my_source_app> --source-app-vault <my_source_app_vault>
```

8. (*Optional*) Vérifiez l'état et l'état de la relation de réplication :

```
kubectl get amr -n my-app-namespace <relationship name> -o=jsonpath
='{.status}' | jq
```

Basculement vers le cluster de destination

À l'aide de Trident Protect, vous pouvez basculer les applications répliquées vers un cluster de destination. Cette procédure arrête la relation de réplication et met l'application en ligne sur le cluster de destination. Trident Protect n'arrête pas l'application sur le cluster source si celle-ci était opérationnelle.

Étapes

1. Ouvrez le fichier CR AppMirrorRelationship (par exemple, `trident-protect-relationship.yaml`) et définissez la valeur de **spec.desiredState** sur `Promoted`.
2. Enregistrez le fichier CR.
3. Appliquer la CR :

```
kubectl apply -f trident-protect-relationship.yaml -n my-app-namespace
```

4. (*Facultatif*) Créez les plannings de protection dont vous avez besoin sur l'application ayant fait l'objet d'un basculement.
5. (*Optional*) Vérifiez l'état et l'état de la relation de réplication :

```
kubectl get amr -n my-app-namespace <relationship name> -o=jsonpath
='{.status}' | jq
```

Resynchronisation d'une relation de réplication ayant échoué

L'opération de resynchronisation rétablit la relation de réplication. Une fois l'opération de resynchronisation effectuée, l'application source d'origine devient l'application en cours d'exécution et toutes les modifications apportées à l'application en cours d'exécution sur le cluster de destination sont supprimées.

Le processus arrête l'application sur le cluster de destination avant de rétablir la réplication.

> Toutes les données écrites sur l'application de destination pendant le basculement sont perdues.

Étapes

1. Créez un instantané de l'application source.
2. Ouvrez le fichier CR AppMirrorRelationship (par exemple, `trident-protect-relationship.yaml`) et définissez la valeur spec.desiredState sur `Established`.
3. Enregistrez le fichier CR.
4. Appliquer la CR :

```
kubectl apply -f trident-protect-relationship.yaml -n my-app-namespace
```

5. Si vous avez créé des plannings de protection sur le cluster de destination pour protéger l'application en panne, supprimez-les. Toute planification qui reste à l'origine de défaillances des snapshots de volume.

Inversion de la resynchronisation d'une relation de réplication ayant échoué

Lorsque vous inversez la resynchronisation d'une relation de réplication ayant fait l'objet d'un basculement, l'application de destination devient l'application source et la source devient la destination. Les modifications apportées à l'application de destination pendant le basculement sont conservées.

Étapes

1. Supprimez la CR AppMirrorRelationship sur le cluster de destination d'origine. La destination devient alors la source. S'il reste des plannings de protection sur le nouveau cluster de destination, supprimez-les.
2. Configurez une relation de réplication en appliquant les fichiers CR que vous avez utilisés à l'origine pour configurer la relation aux clusters opposés.
3. Assurez-vous que les CRS AppVault sont prêts sur chaque cluster.
4. Configurez une relation de réplication sur le cluster opposé, en configurant les valeurs pour la direction inverse.

Inverser le sens de réplication de l'application

Lorsque vous inversez le sens de la réplication, Trident Protect déplace l'application vers le back-end de stockage de destination tout en continuant à répliquer à nouveau vers le back-end de stockage source d'origine. Trident Protect arrête l'application source et réplique les données vers la destination avant de basculer vers l'application cible.

Dans ce cas, vous permutez la source et la destination.

Étapes

1. Créer un instantané d'arrêt :

Créez un instantané d'arrêt à l'aide d'une CR

a. Désactivez les plannings de stratégie de protection pour l'application source.

b. Créer un fichier ShutdownSnapshot CR :

i. Créez le fichier de ressource personnalisée (CR) et nommez-le (par exemple, `trident-protect-shutdownsnapshot.yaml`).

ii. Configurez les attributs suivants :

- **metadata.name**: (*required*) le nom de la ressource personnalisée.
- **Spec.AppVaultRef**: (*required*) cette valeur doit correspondre au champ metadata.name de l'AppVault pour l'application source.
- **Spec.ApplicationRef**: (*required*) cette valeur doit correspondre au champ metadata.name du fichier CR de l'application source.

Exemple YAML :

```
apiVersion: protect.trident.netapp.io/v1
kind: ShutdownSnapshot
metadata:
  name: replication-shutdown-snapshot-afc4c564-e700-4b72-86c3-c08a5dbe844e
  namespace: my-app-namespace
spec:
  appVaultRef: generic-s3-trident-protect-src-bucket-04b6b4ec-46a3-420a-b351-45795e1b5e34
  applicationRef: maria
```

c. Une fois que vous avez rempli le `trident-protect-shutdownsnapshot.yaml` fichier avec les valeurs correctes, appliquez la CR :

```
kubectl apply -f trident-protect-shutdownsnapshot.yaml -n my-app-namespace
```

Créer un snapshot d'arrêt à l'aide de l'interface de ligne de commandes

a. Créez l'instantané d'arrêt, en remplaçant les valeurs entre parenthèses par les informations de votre environnement. Par exemple :

```
tridentctl protect create shutdownsnapshot <my_shutdown_snapshot>
--appvault <my_vault> --app <app_to_snapshot>
```

2. Une fois le snapshot terminé, obtenez l'état du snapshot :

```
kubectl get shutdownsnapshot -n my-app-namespace
<shutdown_snapshot_name> -o yaml
```

3. Recherchez la valeur de **shutdownsnapshot.status.appArchivePath** à l'aide de la commande suivante et enregistrez la dernière partie du chemin d'accès au fichier (également appelée nom de base ; ce sera tout après la dernière barre oblique) :

```
k get shutdownsnapshot -n my-app-namespace <shutdown_snapshot_name> -o
jsonpath='{.status.appArchivePath}'
```

4. Effectuez un basculement du cluster de destination vers le cluster source, avec la modification suivante :

> ⓘ À l'étape 2 de la procédure de basculement, incluez le `spec.promotedSnapshot` champ dans le fichier CR AppMirrorRelationship et définissez sa valeur sur le nom de base que vous avez enregistré à l'étape 3 ci-dessus.

5. Effectuez les étapes de resynchronisation inverse dans Inversion de la resynchronisation d'une relation de réplication ayant échoué.
6. Activez les plannings de protection sur le nouveau cluster source.

Résultat

Les actions suivantes se produisent en raison de la réplication inverse :

- Une copie Snapshot des ressources Kubernetes de l'application source d'origine est effectuée.
- Les pods de l'application source d'origine sont « interrompus » en supprimant les ressources Kubernetes de l'application (laissant les demandes de volume persistant et les volumes persistants en place).
- Une fois les pods arrêtés, des copies Snapshot des volumes de l'application sont prises et répliquées.
- Les relations SnapMirror sont rompues, les volumes de destination étant prêts pour la lecture/l'écriture.
- Les ressources Kubernetes de l'application sont restaurées à partir du snapshot de pré-arrêt, à l'aide des données du volume répliquées après la fermeture de l'application source d'origine.
- La réplication est rétablie dans la direction inverse.

Rétablir le fonctionnement des applications sur le cluster source d'origine

Grâce à Trident Protect, vous pouvez obtenir le « retour arrière » après un basculement en suivant la séquence suivante. Dans ce flux de travail pour restaurer le sens de réplication d'origine, Trident Protect réplique (resyncs) toute modification d'application vers l'application source d'origine avant d'inverser le sens de réplication.

Ce processus commence à partir d'une relation qui a effectué un basculement vers une destination et implique les étapes suivantes :

- Commencer par un état de basculement défaillant.
- Resynchronisez la relation de réplication en sens inverse.

N'effectuez pas d'opération de resynchronisation normale, car cela vous permettra d'ignorer les données écrites sur le cluster de destination pendant la procédure de basculement.

- Inversez le sens de réplication.

Étapes

1. Effectuer les Inversion de la resynchronisation d'une relation de réplication ayant échoué étapes.
2. Effectuer les Inverser le sens de réplication de l'application étapes.

Supprimer une relation de réplication

Vous pouvez supprimer une relation de réplication à tout moment. Lorsque vous supprimez la relation de réplication d'application, deux applications distinctes n'ont aucune relation entre elles.

Étapes

1. Supprimez la CR d'AppMirrorRelationship :

```
kubectl delete -f trident-protect-relationship.yaml -n my-app-namespace
```

Migrez les applications

Vous pouvez migrer vos applications entre des clusters ou des classes de stockage en restaurant vos données de sauvegarde ou d'instantané sur un autre cluster ou une autre classe de stockage.

Lorsque vous migrez une application, tous les crochets d'exécution configurés pour l'application sont migrés avec l'application. Si un hook d'exécution post-restauration est présent, il s'exécute automatiquement dans le cadre de l'opération de restauration.

Opérations de sauvegarde et de restauration

Pour effectuer des opérations de sauvegarde et de restauration dans les scénarios suivants, vous pouvez automatiser des tâches de sauvegarde et de restauration spécifiques.

Clone dans le même cluster

Pour cloner une application sur le même cluster, créez un Snapshot ou sauvegardez et restaurez les données sur le même cluster.

Étapes

1. Effectuez l'une des opérations suivantes :
   a. "Créer un snapshot".
   b. "Créer une sauvegarde".
2. Sur le même cluster, effectuez l'une des opérations suivantes, selon que vous avez créé un snapshot ou une sauvegarde :
   a. "Restaurez vos données à partir du snapshot".
   b. "Restaurez vos données à partir de la sauvegarde".

Cloner vers un autre cluster

Pour cloner une application sur un autre cluster (effectuez un clone entre clusters), créez un snapshot ou une sauvegarde et restaurez les données sur un autre cluster. Assurez-vous que Trident Protect est installé sur le cluster de destination.

Étapes

1. Effectuez l'une des opérations suivantes :
   a. "Créer un snapshot".
   b. "Créer une sauvegarde".
2. Assurez-vous que la CR AppVault du compartiment de stockage objet contenant la sauvegarde ou le snapshot a été configurée sur le cluster de destination.
3. Sur le cluster de destination, effectuez l'une des opérations suivantes, selon que vous avez créé un snapshot ou une sauvegarde :
   a. "Restaurez vos données à partir du snapshot".
   b. "Restaurez vos données à partir de la sauvegarde".

Migration des applications d'une classe de stockage vers une autre

Vous pouvez migrer des applications d'une classe de stockage vers une autre classe de stockage en restaurant un snapshot sur la classe de stockage de destination différente.

Par exemple (à l'exclusion des secrets de la CR de restauration) :

```
apiVersion: protect.trident.netapp.io/v1
kind: SnapshotRestore
metadata:
  name: "${snapshotRestoreCRName}"
spec:
  appArchivePath: "${snapshotArchivePath}"
  appVaultRef: "${appVaultCRName}"
  namespaceMapping:
    destination: "${destinationNamespace}"
    source: "${sourceNamespace}"
  storageClassMapping:
    destination: "${destinationStorageClass}"
    source: "${sourceStorageClass}"
  resourceFilter:
    resourceMatchers:
      kind: Secret
      version: v1
    resourceSelectionCriteria: exclude
```

Restaurez l'instantané à l'aide d'une CR

1. Créez le fichier de ressource personnalisée (CR) et nommez-le `trident-protect-snapshot-restore-cr.yaml`.
2. Dans le fichier que vous avez créé, configurez les attributs suivants :
   - **metadata.name**: (*required*) le nom de cette ressource personnalisée; choisissez un nom unique et sensible pour votre environnement.
   - **Spec.appArchivePath** : chemin d'accès dans AppVault où sont stockés le contenu de l'instantané. Vous pouvez utiliser la commande suivante pour trouver ce chemin :

     ```
     kubectl get snapshots <my-snapshot-name> -n trident-protect -o jsonpath='{.status.appArchivePath}'
     ```

   - **Spec.appVaultRef**: (*required*) le nom du AppVault dans lequel le contenu de l'instantané est stocké.
   - **spec.namespaceMapping**: mappage de l'espace de noms source de l'opération de restauration sur l'espace de noms de destination. Remplacez `my-source-namespace` et `my-destination-namespace` par des informations provenant de votre environnement.

     ```
     apiVersion: protect.trident.netapp.io/v1
     kind: SnapshotRestore
     metadata:
       name: my-cr-name
       namespace: trident-protect
     spec:
       appArchivePath: my-snapshot-path
       appVaultRef: appvault-name
       namespaceMapping: [{"source": "my-source-namespace",
     "destination": "my-destination-namespace"}]
     ```

3. Si vous avez besoin de sélectionner uniquement certaines ressources de l'application à restaurer, ajoutez un filtrage qui inclut ou exclut les ressources marquées d'étiquettes particulières :
   - **ResourceFilter.resourceSelectionCriteria**: (Requis pour le filtrage) utilisez `include or exclude` pour inclure ou exclure une ressource définie dans resourceMatchers. Ajoutez les paramètres resourceMatchers suivants pour définir les ressources à inclure ou à exclure :
   - **resourceMatchers.group**: (*Optional*) Groupe de la ressource à filtrer.
   - **ResourceMatchers.type**: (*Optional*) Type de la ressource à filtrer.
   - **ResourceMatchers.version**: (*Optional*) version de la ressource à filtrer.
   - **resourceMatchers.names**: (*Optional*) noms dans le champ Kubernetes metadata.name de la ressource à filtrer.
   - **resourceMatchers.namespaces**: (*Optional*) espaces de noms dans le champ Kubernetes metadata.name de la ressource à filtrer.
   - **ResourceMatchers.labelSelectors**: (*Optional*) chaîne de sélecteur de libellé dans le champ

Kubernetes metadata.name de la ressource, comme défini dans le "Documentation Kubernetes". Par exemple : `"trident.netapp.io/os=linux"`.

Par exemple :

```
spec:
  resourceFilter:
    resourceSelectionCriteria: "include"
    resourceMatchers:
      group: my-resource-group
      kind: my-resource-kind
      version: my-resource-version
      names: ["my-resource-names"]
      namespaces: ["my-resource-namespaces"]
      labelSelectors: ["trident.netapp.io/os=linux"]
```

4. Une fois que vous avez rempli le `trident-protect-snapshot-restore-cr.yaml` fichier avec les valeurs correctes, appliquez la CR :

```
kubectl apply -f trident-protect-snapshot-restore-cr.yaml
```

Restaurez le snapshot à l'aide de l'interface de ligne de commande

1. Restaurez l'instantané dans un autre espace de noms, en remplaçant les valeurs entre parenthèses par les informations de votre environnement.

   - L'`snapshot` argument utilise un nom d'espace de noms et un nom d'instantané au format `<namespace>/<name>`.
   - L'`namespace-mapping` argument utilise des espaces de noms séparés par deux-points pour mapper les espaces de noms source aux espaces de noms de destination corrects dans le format `source1:dest1,source2:dest2`.

   Par exemple :

```
tridentctl protect create snapshotrestore <my_restore_name>
--snapshot <namespace/snapshot_to_restore> --namespace-mapping
<source_to_destination_namespace_mapping>
```

Gérer les crochets d'exécution

Un crochet d'exécution est une action personnalisée que vous pouvez configurer pour s'exécuter conjointement avec une opération de protection des données d'une application gérée. Par exemple, si vous disposez d'une application de base de données, vous pouvez utiliser un crochet d'exécution pour suspendre toutes les transactions de

base de données avant un instantané et reprendre les transactions une fois l'instantané terminé. Les snapshots sont ainsi cohérents au niveau des applications.

Types de crochets d'exécution

Trident Protect prend en charge les types de crochets d'exécution suivants, en fonction du moment où ils peuvent être exécutés :

- Pré-instantané
- Post-snapshot
- Avant sauvegarde
- Post-sauvegarde
- Post-restauration
- Après le basculement

Ordre d'exécution

Lors de l'exécution d'une opération de protection des données, les événements de hook d'exécution ont lieu dans l'ordre suivant :

1. Tous les crochets d'exécution de pré-opération personnalisés applicables sont exécutés sur les conteneurs appropriés. Vous pouvez créer et exécuter autant de crochets de pré-opération personnalisés que vous le souhaitez, mais l'ordre d'exécution de ces crochets avant que l'opération ne soit ni garantie ni configurable.
2. L'opération de protection des données est effectuée.
3. Tous les crochets d'exécution de post-opération personnalisés applicables sont exécutés sur les conteneurs appropriés. Vous pouvez créer et exécuter autant de crochets post-opération personnalisés que vous le souhaitez, mais l'ordre d'exécution de ces crochets après l'opération n'est ni garanti ni configurable.

Si vous créez plusieurs crochets d'exécution du même type (par exemple, pré-instantané), l'ordre d'exécution de ces crochets n'est pas garanti. Cependant, l'ordre d'exécution des crochets de différents types est garanti. Par exemple, voici l'ordre d'exécution d'une configuration qui a tous les types de crochets :

1. Crochets pré-instantanés exécutés
2. Crochets post-snapshot exécutés
3. Crochets de pré-secours exécutés
4. Crochets post-secours exécutés

L'exemple d'ordre précédent s'applique uniquement lorsque vous exécutez une sauvegarde qui n'utilise pas de snapshot existant.

Vous devez toujours tester vos scripts d'exécution avant de les activer dans un environnement de production. Vous pouvez utiliser la commande 'kubectl exec' pour tester aisément les scripts. Une fois que vous avez activé les crochets d'exécution dans un environnement de production, testez les snapshots et les sauvegardes obtenus pour vous assurer qu'ils sont cohérents. Pour ce faire, vous pouvez cloner l'application dans un espace de noms temporaire, restaurer le snapshot ou la sauvegarde, puis tester l'application.

Remarques importantes sur les crochets d'exécution personnalisés

Lors de la planification de crochets d'exécution pour vos applications, tenez compte des points suivants.

- Un crochet d'exécution doit utiliser un script pour effectuer des actions. De nombreux crochets d'exécution peuvent référencer le même script.
- Trident Protect exige que les scripts utilisés par les crochets d'exécution soient écrits au format de scripts shell exécutables.
- La taille du script est limitée à 96 Ko.
- Trident Protect utilise les paramètres du crochet d'exécution et les critères correspondants pour déterminer les crochets applicables à une opération de snapshot, de sauvegarde ou de restauration.

> Puisque les crochets d'exécution réduisent ou désactivent complètement la fonctionnalité de l'application contre laquelle ils s'exécutent, vous devez toujours essayer de réduire le temps d'exécution de vos crochets personnalisés. Si vous démarrez une opération de sauvegarde ou d'instantané avec les crochets d'exécution associés, mais que vous l'annulez, les crochets sont toujours autorisés à s'exécuter si l'opération de sauvegarde ou d'instantané a déjà commencé. Cela signifie que la logique utilisée dans un crochet d'exécution post-sauvegarde ne peut pas présumer que la sauvegarde a été effectuée.

Filtres de crochet d'exécution

Lorsque vous ajoutez ou modifiez un crochet d'exécution pour une application, vous pouvez ajouter des filtres au crochet d'exécution pour gérer les conteneurs auxquels le crochet correspond. Les filtres sont utiles pour les applications qui utilisent la même image de conteneur sur tous les conteneurs, mais ils peuvent utiliser chaque image à des fins différentes (comme Elasticsearch). Les filtres vous permettent de créer des scénarios dans lesquels des crochets d'exécution s'exécutent sur certains conteneurs, mais pas nécessairement tous identiques. Si vous créez plusieurs filtres pour un seul crochet d'exécution, ils sont combinés avec un opérateur ET logique. Vous pouvez avoir jusqu'à 10 filtres actifs par crochet d'exécution.

Chaque filtre que vous ajoutez à un crochet d'exécution utilise une expression régulière pour faire correspondre les conteneurs de votre cluster. Lorsqu'un crochet correspond à un conteneur, le crochet exécute son script associé sur ce conteneur. Les expressions régulières pour les filtres utilisent la syntaxe de l'expression régulière 2 (RE2), qui ne prend pas en charge la création d'un filtre qui exclut les conteneurs de la liste des correspondances. Pour plus d'informations sur la syntaxe prise en charge par Trident Protect pour les expressions régulières dans les filtres de hook d'exécution, reportez-vous àla section "Prise en charge de la syntaxe de l'expression régulière 2 (RE2)".

> Si vous ajoutez un filtre d'espace de noms à un crochet d'exécution qui s'exécute après une opération de restauration ou de clonage et que la source et la destination de restauration ou de clonage sont dans des espaces de noms différents, le filtre d'espace de noms est appliqué uniquement à l'espace de noms de destination.

Exemples de crochet d'exécution

Visitez le "Projet GitHub NetApp Verda" pour télécharger des scripts d'exécution réels pour les applications courantes telles qu'Apache Cassandra et Elasticsearch. Vous pouvez également voir des exemples et obtenir des idées pour structurer vos propres crochets d'exécution personnalisés.

Créer un crochet d'exécution

Vous pouvez créer un crochet d'exécution personnalisé pour une application à l'aide de Trident Protect. Vous

devez disposer d’autorisations propriétaire, administrateur ou membre pour créer des crochets d’exécution.

Utiliser une CR

1. Créez le fichier de ressource personnalisée (CR) et nommez-le `trident-protect-hook.yaml`.
2. Configurez les attributs suivants en fonction de votre environnement Trident Protect et de la configuration du cluster :

   - **metadata.name**: (*required*) le nom de cette ressource personnalisée; choisissez un nom unique et sensible pour votre environnement.
   - **Spec.applicationRef**: (*required*) Nom Kubernetes de l'application pour laquelle exécuter le hook d'exécution.
   - **Spec.stage**: (*required*) Une chaîne indiquant quelle étape de l'action doit être exécutée par le crochet d'exécution. Valeurs possibles :
     - Pré
     - Post
   - **Spec.action**: (*required*) Une chaîne indiquant l'action que prendra le crochet d'exécution, en supposant que tous les filtres de crochet d'exécution spécifiés soient mis en correspondance. Valeurs possibles :
     - Snapshot
     - Sauvegarde
     - Restaurer
     - Basculement
   - **Spec.enabled**: (*Optional*) indique si ce hook d'exécution est activé ou désactivé. Si elle n'est pas spécifiée, la valeur par défaut est true.
   - **Spec.hookSource**: (*required*) chaîne contenant le script hook codé en base64.
   - **Spec.timeout**: (*Optional*) nombre définissant la durée en minutes pendant laquelle le crochet d'exécution est autorisé à s'exécuter. La valeur minimale est de 1 minute et la valeur par défaut est de 25 minutes si elle n'est pas spécifiée.
   - **Spec.arguments**: (*Optional*) liste YAML d'arguments que vous pouvez spécifier pour le crochet d'exécution.
   - **Spec.matchingCriteria**: (*Optional*) liste facultative de paires de valeurs de clé de critères, chaque paire constituant un filtre de crochet d'exécution. Vous pouvez ajouter jusqu'à 10 filtres par crochet d'exécution.
   - **Spec.matchingCriteria.type**: (*Optional*) chaîne identifiant le type de filtre du crochet d'exécution. Valeurs possibles :
     - ContainerImage
     - ContainerName
     - PodName
     - PodLabel
     - NomespaceName
   - **Spec.matchingCriteria.Value**: (*Optional*) Une chaîne ou Une expression régulière identifiant la valeur du filtre crochet d'exécution.

     Exemple YAML :

```
apiVersion: protect.trident.netapp.io/v1
kind: ExecHook
metadata:
  name: example-hook-cr
  namespace: my-app-namespace
  annotations:
    astra.netapp.io/astra-control-hook-source-id:
/account/test/hookSource/id
spec:
  applicationRef: my-app-name
  stage: Pre
  action: Snapshot
  enabled: true
  hookSource: IyEvYmluL2Jhc2gKZWNobyAiZXhhbXBsZSBzY3JpcHQiCg==
  timeout: 10
  arguments:
    - FirstExampleArg
    - SecondExampleArg
  matchingCriteria:
    - type: containerName
      value: mysql
    - type: containerImage
      value: bitnami/mysql
    - type: podName
      value: mysql
    - type: namespaceName
      value: mysql-a
    - type: podLabel
      value: app.kubernetes.io/component=primary
    - type: podLabel
      value: helm.sh/chart=mysql-10.1.0
    - type: podLabel
      value: deployment-type=production
```

3. Une fois que vous avez rempli le fichier CR avec les valeurs correctes, appliquez la CR :

```
kubectl apply -f trident-protect-hook.yaml
```

Utiliser l'interface de ligne de commande

1. Créez le crochet d'exécution en remplaçant les valeurs entre parenthèses par les informations de votre environnement. Par exemple :

```
tridentctl protect create exechook <my_exec_hook_name> --action
<action_type> --app <app_to_use_hook> --stage <pre_or_post_stage>
--source-file <script-file>
```

Désinstallez Trident Protect

Vous devrez peut-être supprimer les composants Trident Protect si vous effectuez une mise à niveau d'une version d'évaluation vers une version complète du produit.

Pour supprimer Trident Protect, effectuez les opérations suivantes.

Étapes

1. Supprimez les fichiers Trident Protect CR :

```
helm uninstall trident-protect-crds
```

2. Supprimer Trident Protect :

```
helm uninstall -n trident-protect trident-protect
```

3. Supprimez l'espace de noms Trident Protect :

```
kubectl delete ns trident-protect
```

Connaissances et support

Foire aux questions

Trouvez les réponses aux questions fréquemment posées sur l'installation, la configuration, la mise à niveau et le dépannage de Trident.

Questions générales

À quelle fréquence le Trident est-il commercialisé ?

Trident est commercialisé tous les quatre mois à partir de la version 24.02 : février, juin et octobre.

Trident prend-il en charge toutes les fonctionnalités publiées dans une version spécifique de Kubernetes ?

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.

Le fonctionnement de Trident dépend-il d'autres produits NetApp ?

Trident ne dépend d'aucun autre produit logiciel NetApp et fonctionne comme une application autonome. Toutefois, vous devez disposer d'un système de stockage back-end NetApp.

Comment puis-je obtenir des informations complètes sur la configuration de Trident ?

Utilisez `tridentctl get` la commande pour obtenir plus d'informations sur votre configuration Trident.

Puis-je obtenir des indicateurs sur le provisionnement du stockage par Trident ?

Oui. Les terminaux Prometheus peuvent être utilisés pour rassembler des informations sur les opérations Trident, telles que le nombre de systèmes back-end gérés, le nombre de volumes provisionnés, d'octets consommés, etc. Vous pouvez également l'utiliser "Cloud Insights" pour la surveillance et l'analyse.

L'expérience utilisateur change-t-elle lors de l'utilisation de Trident en tant que mécanisme de provisionnement CSI ?

Non. Il n'y a pas de changement en ce qui concerne l'expérience utilisateur et les fonctionnalités. Le nom de provisionneur utilisé est `csi.trident.netapp.io`. Cette méthode d'installation de Trident est recommandée si vous souhaitez utiliser toutes les nouvelles fonctionnalités fournies par les versions actuelles et futures.

Installez et utilisez Trident sur un cluster Kubernetes

Trident prend-il en charge une installation hors ligne à partir d'un registre privé ?

Oui, Trident peut être installé hors ligne. Reportez-vous à la "En savoir plus sur l'installation de Trident".

Puis-je installer Trident à distance ?

Oui. Trident 18.10 et les versions ultérieures prennent en charge la fonctionnalité d'installation à distance à partir de n'importe quelle machine ayant `kubectl` accès au cluster. Une fois l' `kubectl`accès vérifié (par exemple, lancez une `kubectl get nodes` commande à partir de l'ordinateur distant pour vérifier), suivez les instructions d'installation.

Puis-je configurer la haute disponibilité avec Trident ?

Trident est installé en tant que déploiement Kubernetes (ReplicaSet) avec une instance. Sa haute disponibilité est donc intégrée. Vous ne devez pas augmenter le nombre de répliques dans le déploiement. Si le nœud sur lequel Trident est installé est perdu ou si le pod est inaccessible, Kubernetes redéploie automatiquement le pod vers un nœud sain dans le cluster. Trident est un plan de contrôle uniquement. Les pods actuellement montés ne sont donc pas affectés si Trident est redéployé.

Trident a-t-il besoin d'accéder à l'espace de noms du système kube ?

Trident lit depuis le serveur d'API Kubernetes pour déterminer quand les applications demandent de nouvelles ESV et doit donc accéder à kube-system.

Quels sont les rôles et les Privileges utilisés par Trident ?

Le programme d'installation Trident crée un cluster Kubernetes ClusterRole, qui dispose d'un accès spécifique aux ressources PersistentVolume, PersistentVolumeClaim, StorageClass et Secret du cluster Kubernetes. Reportez-vous à la section "Personnalisez l'installation tridentctl".

Puis-je générer localement les fichiers de manifeste exacts que Trident utilise pour l'installation ?

Vous pouvez, si nécessaire, générer et modifier localement les fichiers de manifeste exacts que Trident utilise pour l'installation. Reportez-vous à la "Personnalisez l'installation tridentctl".

Puis-je partager le même SVM back-end ONTAP pour deux instances Trident distinctes pour deux clusters Kubernetes distincts ?

Bien qu'il ne soit pas conseillé, vous pouvez utiliser la même SVM back-end pour deux instances Trident. Spécifiez un nom de volume unique pour chaque instance lors de l'installation et/ou spécifiez un paramètre unique `StoragePrefix` dans `setup/backend.json` le fichier. Ceci permet de s'assurer que le même FlexVol n'est pas utilisé pour les deux instances.

Est-il possible d'installer Trident sous ContainerLinux (anciennement CoreOS) ?

Trident est un pod Kubernetes qui peut être installé quel que soit l'emplacement d'exécution de Kubernetes.

Puis-je utiliser Trident avec NetApp Cloud Volumes ONTAP ?

Oui, Trident est pris en charge sur AWS, Google Cloud et Azure.

Trident fonctionne-t-il avec Cloud volumes Services ?

Oui, Trident prend en charge le service Azure NetApp Files dans Azure ainsi que Cloud Volumes Service dans GCP.

Dépannage et support

NetApp prend-il en charge Trident ?

Bien que Trident soit open source et gratuit, NetApp le prend entièrement en charge, à condition que votre 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 ?

Créez un problème "Trident Github" et mentionnez **RFE** dans l'objet et la description du problème.

Où puis-je soulever un défaut ?

Créez un problème sur "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 Trident et que j'ai besoin d'éclaircissements ? Y a-t-il une communauté ou un forum?

Pour toute question, problème ou demande, contactez-nous par le biais de notre Trident "Déroulez le canal"ou GitHub.

Le mot de passe de mon système de stockage a changé et Trident ne fonctionne plus. Comment puis-je le récupérer ?

Mettez à jour le mot de passe du back-end 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.

Trident ne trouve pas mon nœud Kubernetes. Comment résoudre ce problème ?

Trident ne trouve pas de nœud Kubernetes dans deux scénarios possibles. 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 se sont produites après l'installation de Trident, ce problème survient uniquement avec les nouveaux nœuds Kubernetes 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 Trident sont stockées dans des objets CRD. Tous les volumes persistants provisionnés par Trident fonctionneront normalement.

Mettez à niveau 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 de Trident d'une version majeure vers la prochaine version majeure immédiate. 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 ?

Si vous avez besoin d'un correctif pour les bogues observés après une mise à niveau, des problèmes de dépendance ou une mise à niveau non réussie ou incomplète, vous devez "Désinstallez Trident"réinstaller la version précédente en suivant les instructions spécifiques à cette version. Il s'agit de la seule méthode recommandée pour revenir à une version antérieure.

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. Trident utilise "Mappage de LUN sélectif ONTAP" pour découvrir les LIF iSCI nécessaires à l'établissement d'une session à chemins multiples. Un avertissement est généré si `dataLIF` est explicitement défini. Voir "Options et exemples de configuration des SAN ONTAP" pour plus de détails.
- NAS ONTAP : nous vous recommandons de spécifier `dataLIF`. Si non fourni, Trident récupère les LIFs 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. Voir "Options et exemples de configuration du NAS ONTAP"pour plus de détails

Trident peut-il configurer CHAP pour les systèmes back-end ONTAP ?

Oui. Trident prend en charge le protocole CHAP bidirectionnel pour les systèmes back-end ONTAP. Ceci nécessite la `useCHAP=true` configuration de votre back-end.

Comment gérer les règles d'exportation avec Trident ?

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. De cette manière, Trident gère automatiquement l'ajout et la suppression de règles pour les nœuds avec des adresses IP dans les délais de modification donnés.

Les adresses IPv6 peuvent-elles être utilisées pour les LIF de données et de gestion ?

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.

Trident doit être installé à l'aide de l'indicateur `--use-ipv6` (pour l' `tridentctl`installation), `IPv6` (pour l'opérateur Trident) ou `tridentTPv6` (pour l'installation Helm) 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 Trident pour Kubernetes ?

Trident peut prendre en charge plusieurs systèmes back-end simultanément, avec le même pilote ou des pilotes différents.

Comment Trident stocke-t-il les informations d'identification back-end ?

Trident stocke les informations d'identification du back-end en tant que secrets Kubernetes.

Comment Trident sélectionne-t-il un 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 que Trident ne se provisionne pas à partir d'un back-end spécifique ?

Le `excludeStoragePools` paramètre est utilisé pour filtrer l'ensemble de pools que Trident utilise pour le provisionnement et supprime tous les pools correspondant.

En cas de systèmes back-end multiples du même type, comment Trident sélectionne-t-il le système back-end à utiliser ?

S'il existe plusieurs systèmes back-end configurés du même type, Trident sélectionne le back-end approprié en fonction des paramètres présents dans `StorageClass` et `PersistentVolumeClaim`. Par exemple, s'il existe plusieurs systèmes back-end de pilotes ONTAP-nas, Trident tente de faire correspondre les paramètres dans le `StorageClass` et le combiné et `PersistentVolumeClaim` un back-end qui peut répondre aux exigences répertoriées dans `StorageClass` le et `PersistentVolumeClaim`le . Si plusieurs systèmes back-end correspondent à la demande, Trident les sélectionne de manière aléatoire.

Trident prend-il en charge le protocole CHAP bidirectionnel avec Element/SolidFire ?

Oui.

Comment Trident déploie-t-il les 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 Trident en définissant un paramètre dans le fichier de définition back-end.

Comment configurer un ensemble explicite d'options de montage NFS ONTAP lors du provisionnement d'un volume ?

Par défaut, Trident ne définit aucune valeur des options de montage sur 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 des volumes via Trident avec 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, reportez-vous à : "Fonctionnement de Trident avec NVE et NAE"

Quel est le meilleur moyen d'implémenter la QoS pour ONTAP via Trident ?

Utiliser `StorageClasses` Afin d'implémenter la QoS pour ONTAP.

Comment spécifier le provisionnement fin ou non fin via 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 ESV NFS créées par Trident ?

Oui. Vous pouvez développer une demande de volume persistant créée par Trident. Notez que la croissance automatique de volume est une fonctionnalité ONTAP qui n'est pas applicable à Trident.

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.
```

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 des quotas Kubernetes en raison d'un manque de capacité, Trident tente de se provisionner, mais des erreurs se produisent.

Puis-je créer des copies Snapshot de volume à l'aide de Trident ?

Oui. La création de snapshots de volumes à la demande et de volumes persistants à partir de snapshots sont prises en charge par Trident. Pour créer des volumes persistants à partir de snapshots, assurez-vous que la `VolumeSnapshotDataSource` porte de fonctionnalité a été activée.

Quels sont les pilotes qui prennent en charge les copies Snapshot de volume 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 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.

Cela est également disponible sur les `ontap-nas-economy` pilotes, mais au niveau de la granularité FlexVol, et non au niveau qtree. Pour activer la fonction de snapshot des volumes provisionnés par Trident, définissez l'option du paramètre back-end `snapshotPolicy` sur la règle de snapshot souhaitée, comme défini sur le back-end ONTAP. Les snapshots pris par le contrôleur de stockage ne sont pas connus par Trident.

Puis-je définir un pourcentage de réserve Snapshot pour un volume provisionné via Trident ?

Oui, vous pouvez réserver un pourcentage spécifique d'espace disque pour stocker les copies Snapshot via Trident en définissant l' `snapshotReserve`attribut dans le fichier de définition back-end. Si vous avez configuré `snapshotPolicy` et `snapshotReserve` dans le fichier de définition back-end, le pourcentage de réserve de snapshots est défini en fonction du `snapshotReserve` pourcentage mentionné dans le fichier back-end. Si le `snapshotReserve` pourcentage n'est pas mentionné, ONTAP utilise

par défaut le pourcentage de réserve d'instantanés à 5. Si l' `snapshotPolicy`option est définie sur aucun, le pourcentage de réserve d'instantanés 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 les volumes via 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.

Dépannage

Utilisez les pointeurs fournis ici pour résoudre les problèmes que vous pouvez rencontrer lors de l'installation et de l'utilisation de Trident.

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 Trident 20.07 et 20.10, utiliser à la `tprov` 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 de la configuration requise pour 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 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 effectuer une mise à niveau vers 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.
- 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.

Échec du déploiement de Trident avec 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.

Échec du déploiement de Trident avec `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.
```

Retirez complètement les Trident et les CRD

Vous pouvez supprimer complètement Trident et tous les CRD créés et les ressources personnalisées associées.

Cette opération ne peut pas être annulée. Ne le faites pas à moins que vous ne souhaitiez une installation entièrement nouvelle de Trident. Pour désinstaller Trident sans supprimer les CRD, reportez-vous "Désinstaller Trident"à la section .

Opérateur Trident

Pour désinstaller Trident et supprimer complètement les CRD à l'aide de l'opérateur Trident :

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

Gouvernail

Pour désinstaller Trident et supprimer complètement les CRD à l'aide de l'assistant :

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

<code>tridentctl</code>

Pour supprimer complètement les CRD après avoir désinstallé Trident à l'aide de `tridentctl`

```
tridentctl obliviate crd
```

Échec de l'annulation du transfert de nœud NVMe avec les espaces de noms de bloc bruts RWX o Kubernetes 1.26

Si vous exécutez Kubernetes 1.26, l'annulation de l'environnement de nœud peut échouer lors de l'utilisation de NVMe/TCP avec les espaces de noms de bloc bruts RWX. Les scénarios suivants offrent une solution de contournement à la défaillance. Vous pouvez également mettre à niveau Kubernetes vers la version 1.27.

Espace de noms et pod supprimés

Imaginez un namespace géré par Trident (volume persistant NVMe) attaché à un pod. Si vous supprimez l'espace de nom directement du back-end ONTAP, le processus de déstaging est bloqué après la tentative de suppression du pod. Ce scénario n'a aucun impact sur le cluster Kubernetes ou tout autre fonctionnement.

Solution de contournement

Démontez le volume persistant (correspondant à cet espace de noms) du nœud respectif et supprimez-le.

DataLIFs bloquées

```
 If you block (or bring down) all the dataLIFs of the NVMe Trident
backend, the unstaging process gets stuck when you attempt to delete the
pod. In this scenario, you cannot run any NVMe CLI commands on the
Kubernetes node.
.Solution de contournement
Afficher les dataLIFS pour restaurer toutes les fonctionnalités.
```

Mappage de l'espace de noms supprimé

```
 If you remove the `hostNQN` of the worker node from the corresponding
subsystem, the unstaging process gets stuck when you attempt to delete the
pod. In this scenario, you cannot run any NVMe CLI commands on the
Kubernetes node.
.Solution de contournement
Ajoutez le `hostNQN` retour au sous-système.
```

Assistance

NetApp prend en charge Trident de plusieurs façons. De nombreuses options d'auto-assistance gratuites sont disponibles 24 h/24 et 7 j/7, comme des articles de la base de connaissances (KB) et un canal discorde.

Cycle de vie du support Trident

Trident propose trois niveaux de support en fonction de votre version. Reportez-vous à la "Prise en charge de la version du logiciel NetApp pour les définitions".

Support complet
: Trident offre un support complet pendant douze mois à compter de sa date de sortie.

Prise en charge limitée
: Trident offre un support limité pour les mois 13 à 24 à compter de la date de sortie.

Auto-assistance
: La documentation Trident est disponible pour les mois 25 à 36 à compter de la date de publication.

Version	Support complet	Prise en charge limitée	Auto-assistance
"24,10"	Octobre 2025	Octobre 2026	Octobre 2027
"24.06"	Juin 2025	Juin 2026	Juin 2027
"24.02"	Février 2025	Février 2026	Février 2027
"23.10"	Octobre 2024	Octobre 2025	Octobre 2026

Version	Support complet	Prise en charge limitée	Auto-assistance
"23.07"	Juillet 2024	Juillet 2025	Juillet 2026
"23.04"	—	Avril 2025	Avril 2026
"23.01"	—	Janvier 2025	Janvier 2026
"22.10"	—	Octobre 2024	Octobre 2025
"22.07"	—	Juillet 2024	Juillet 2025
"22.04"	—	—	Avril 2025
"22.01"	—	—	Janvier 2025

Auto-assistance

Pour obtenir une liste complète des articles de dépannage, reportez-vous à "Base de connaissances NetApp (identifiant requis)"la .

Soutien de la communauté

Il existe une communauté publique dynamique d'utilisateurs de conteneurs (y compris les développeurs Trident) sur notre "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.

Support technique NetApp

Pour obtenir de l'aide sur Trident, créez un bundle de support à l'aide de `tridentctl logs -a -n trident` et envoyez-le à `NetApp Support <Getting Help>`.

Pour en savoir plus

- "Ressources Trident"
- "Kubernetes Hub"

Référence

Ports Trident

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

Ports Trident

Trident communique sur les ports suivants :

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

Le port de la sonde 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 Trident

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

Quand utiliser l'API REST

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

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

Avec l'API REST

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

L'API fonctionne comme suit :

OBTENEZ

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

Répertorie tous les objets de ce type.

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

Obtient les détails de l'objet nommé.

POST

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

Crée un objet du type spécifié.

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

SUPPRIMER

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

Supprime la ressource nommée.

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

Options de ligne de commande

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

Journalisation

-debug

Active la sortie de débogage.

-loglevel <level>

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

Kubernetes

-k8s_pod

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

-k8s_api_server <insecure-address:insecure-port>

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

k8s_pod`option.

Docker

-volume_driver <name>

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

-driver_port <port-number>

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

-config <file>

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

REPOS

-address <ip-or-host>

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

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

-rest

Active l'interface REST. Valeur true par défaut.

Kubernetes et objets Trident

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

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

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

1. Un utilisateur crée 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
trident.netapp.io/cloneFromPVC	Volume cloneSourceVolume	ontap-nas, ontap-san, solidfire-san, azure-netapp-files, gcp-cvs ontap-san-économie
trident.netapp.io/splitOnClone	SplitOnClone	ontap-nas, ontap-san
trident.netapp.io/protocol	protocole	toutes
trident.netapp.io/exportPolicy	ExportPolicy	ontap-nas, économie ontap-nas, ontap-nas-flexgroup
trident.netapp.io/snapshotPolicy	Politique de snapshots	ontap-nas, ontap-nas-économie, ontap-nas-flexgroup, ontap-san
trident.netapp.io/snapshotReserve	Réserve de snapshots	ontap-nas, ontap-nas-flexgroup, ontap-san, gcp-cvs

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

Si le volume persistant créé est 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.

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. Reportez-vous à la section Pour obtenir une description complète des paramètres et des paramètres

associés aux volumes 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
Type de provisionnement	chaîne	fin, épais	Le pool prend en charge cette méthode de provisionnement	Méthode de provisionnement spécifiée	thick : tous les systèmes ONTAP ; thin : tous les systèmes ONTAP et solidfire-san
Type de dos	chaîne	ontap-nas, économie ontap-nas, ontap-nas-flexgroup, ontap-san, solidfire-san, gcp-cvs, azure-netapp-files, ontap-san-economy	Le pool appartient à ce type de système back-end	Backend spécifié	Tous les conducteurs

Attribut	Type	Valeurs	Offre	Demande	Pris en charge par
snapshots	bool	vrai, faux	Le pool prend en charge les volumes dotés de snapshots	Volume sur lequel les snapshots sont activés	ontap-nas, ontap-san, solidfire-san, gcp-cvs
clones	bool	vrai, faux	Le pool prend en charge les volumes de clonage	Volume sur lequel les clones sont activés	ontap-nas, ontap-san, solidfire-san, gcp-cvs
le cryptage	bool	vrai, faux	Le pool prend en charge les volumes chiffrés	Volume avec chiffrement activé	ontap-nas, économie ontap-nas, ontap-nas-flexgroups, ontap-san
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	Type de système de fichiers pour les volumes en mode bloc	solidfire-san, ontap-nas, ontap-nas-économie, ontap-nas-flexgroup, ontap-san, ontap-san-économie	Tout
Volumeallowexpansion	booléen	vrai, faux	Activez ou désactivez la prise en charge pour augmenter la taille de la demande de volume persistant	ontap-nas, économie ontap-nas, ontap-nas-flexgroup, ontap-san, ontap-san-économie, solidfire-san, gcp-cvs, azure-netapp-files	1.11+
Volume Bindingmode	chaîne	Immédiat, WaitForFirstConsumer	Sélectionnez le moment où la liaison des volumes et le provisionnement dynamique se produisent	Tout	1.19 - 1.26

- Le `fsType` Paramètre permet de contrôler le type de système de fichiers souhaité pour les LUN SAN. 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. Reportez-vous à la section "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.

- Reportez-vous à la section "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.

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.

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

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

Les CRDS sont une construction Kubernetes. Les objets des ressources définies ci-dessus sont créés par Trident. 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 désinstallation de Trident à l'aide de `tridentctl uninstall` Les pods Trident sont supprimés, mais les CRD créés ne sont pas nettoyés. Reportez-vous à la section "Désinstaller Trident" Afin de comprendre comment Trident peut être entièrement supprimé et reconfiguré de zéro.

ObjetsTrident `StorageClass`

Trident crée des classes de stockage correspondantes pour Kubernetes `StorageClass` objets spécifiés `csi.trident.netapp.io` 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".

ObjetsTrident `StoragePool`

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

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

ObjetsTrident `Volume`

Les volumes 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
SplitOnClone	chaîne	non	ONTAP (nas, san) : séparer le clone de son parent
Politique de snapshots	chaîne	non	ONTAP-* : stratégie d'instantané à utiliser
Réserve de snapshots	chaîne	non	ONTAP-* : pourcentage de volume réservé pour les snapshots
ExportPolicy	chaîne	non	ontap-nas* : export policy à utiliser

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

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.

ObjetsTrident `Snapshot`

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

Chacun `Snapshot` l'objet inclut les propriétés répertoriées ci-dessous :

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

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

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

ObjetTrident `ResourceQuota`

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

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

Si vous avez besoin de plus de contrôle sur le quota de ressources par défaut et la classe de priorité, vous pouvez générer un `custom.yaml` ou 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, voir "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 votre propre allocation de ressources, vous pouvez supprimer l'objet Trident `ResourceQuota` à l'aide de la commande :

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

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

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

PSS remplace les politiques de sécurité Pod (PSP). La PSP est obsolète dans Kubernetes v1.21 et elle sera supprimée dans la version 1.25. Pour plus d'informations, reportez-vous à la section "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, reportez-vous à la section "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. Trident interroge les processus hôtes pour déterminer si `rpc-statd` est en cours d'exécution avant le montage des volumes NFS.
Capacités	Le `SYS_ADMIN` la fonctionnalité fait partie des fonctionnalités par défaut pour les conteneurs privilégiés. Par exemple, Docker définit ces fonctionnalités pour les conteneurs privilégiés : `CapPrm: 0000003fffffffff` `CapEff: 0000003fffffffff`

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

Normes de sécurité du pod (PSS)

Étiquette	Description	Valeur par défaut

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

> La modification des étiquettes de l'espace de noms peut entraîner l'absence de planification des modules, un "erreur de création: …" ou un "avertissement: trident-csi-…". 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

Champ	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
`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`

Champ	Description	Valeur par défaut
`requiredDropCapabilities`	Les pods de nœuds Trident exécutent un conteneur privilégié et ne peuvent pas supprimer de fonctionnalités.	`none`
`runAsGroup`	Les conteneurs Trident s'exécutent en tant que root.	`RunAsAny`
`runAsUser`	Les conteneurs Trident s'exécutent en tant que root.	`runAsAny`
`runtimeClass`	Trident n'utilise pas `RuntimeClasses`.	Vide
`seLinux`	Trident 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`

Étiquettes	Description	Valeur par défaut
`allowedUnsafeSysctls`	Trident n'exige aucun niveau de sécurité `sysctls`.	`none`
`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`

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 droits d’auteur et les licences de tiers utilisés dans le logiciel NetApp pour Trident dans le fichier des avis pour chaque version à l’adresse https://github.com/NetApp/trident/.

Informations sur le copyright

Informations sur les marques commerciales