Documentation Trident 26.06

Trident

NetApp
July 01, 2026

This PDF was generated from https://docs.netapp.com/fr-fr/trident/index.html on July 01, 2026. Always check docs.netapp.com for the latest.

Sommaire

Documentation Trident 26.06 . . . 1
Notes de mise à jour . . . 2
Quoi de neuf . . . 2
Quoi de neuf dans la version 26.06 . . . 2
Nouveautés de la version 26.02.1 . . . 5
Quoi de neuf dans la version 26.02 . . . 6
Quoi de neuf dans la version 25.10 . . . 9
Modifications apportées le 25/06/2020 . . . 11
Modifications dans la version 25.06.1 . . . 11
Modifications dans la version 25.06 . . . 12
Changements en 25.02.1 . . . 14
Changements en 25.02 . . . 14
Changements en 24.10.1 . . . 16
Changements en 24.10 . . . 16
Changements en 24.06 . . . 18
Changements en 24.02 . . . 19
Changements en 23.10 . . . 19
Changements en 23.07.1 . . . 20
Changements en 23.07 . . . 20
Changements en 23.04 . . . 21
Changements en 23.01.1 . . . 22
Changements en 23.01 . . . 23
Changements en 22.10 . . . 23
Changements en 22.07 . . . 25
Changements en 22.04 . . . 26
Changements en 22.01.1 . . . 26
Changements en 22.01.0 . . . 27
Changements en 21.10.1 . . . 27
Changements en 21.10.0 . . . 28
Problèmes connus . . . 28
Trouvez plus d’informations . . . 30
Versions antérieures de la documentation . . . 30
NetApp Trident prise en charge de Trident pour les systèmes de stockage ONTAP ASA r2 . . . 30
Opérations prises en charge . . . 31
Opérations non prises en charge . . . 31
Problèmes connus . . . 31
VolumeSnapshots n’atteignent pas l’état ReadyToUse . . . 31
La restauration des sauvegardes Restic de fichiers volumineux peut échouer . . . 31
Commencez . . . 33
Découvrez Trident . . . 33
Découvrez Trident . . . 33
Architecture Trident . . . 34
Concepts . . . 37

Démarrage rapide pour Trident . . . 41
Et la suite ? . . . 42
De formation . . . 42
Informations critiques sur Trident . . . 42
Systèmes front-end (orchestrateurs) pris en charge . . . 43
Systèmes back-end pris en charge (stockage) . . . 43
Prise en charge de Trident pour KubeVirt et OpenShift Virtualization . . . 43
Prise en charge de Trident pour Kubernetes sur Kubernetes et les plans de contrôle hébergés OpenShift . . . 44
Configuration requise . . . 45
Systèmes d'exploitation hôtes testés . . . 45
Configuration de l'hôte . . . 46
Configuration du système de stockage . . . 46
Ports Trident . . . 46
Images de conteneur et versions Kubernetes correspondantes . . . 46
Installation de Trident . . . 48
En savoir plus sur l'installation de Trident . . . 48
Informations essentielles concernant Trident 26.02 . . . 48
Avant de commencer . . . 48
Choisissez votre méthode d'installation . . . 49
Choisissez votre mode d'installation . . . 51
Sélectionnez le processus en fonction de votre méthode et de votre mode . . . 51
Passage d'une méthode d'installation à l'autre . . . 52
Autres options de configuration connues . . . 52
Installer à l'aide de l'opérateur Trident . . . 52
Déployer manuellement l'opérateur Trident (mode Standard) . . . 52
Déploiement manuel de l'opérateur Trident (mode hors ligne) . . . 58
Déploiement de l'opérateur Trident à l'aide de Helm (mode standard) . . . 64
Déploiement de l'opérateur Trident à l'aide de Helm (mode hors ligne) . . . 72
Personnalisez l'installation de l'opérateur Trident . . . 81
Installation à l'aide de tridentctl . . . 93
Installation à l'aide de tridentctl . . . 93
Personnalisez l'installation tridentctl . . . 97
Installation à l'aide d'un opérateur certifié OpenShift . . . 98
Installez Trident à l'aide d'OpenShift OperatorHub . . . 98
Passez à l'opérateur Trident certifié OpenShift . . . 101
Utilisez Trident . . . 105
Préparez le nœud de travail . . . 105
Choisir les bons outils . . . 105
Détection des services de nœud . . . 105
Volumes NFS . . . 106
Volumes iSCSI . . . 106
Volumes NVMe/TCP . . . 110
Volumes SCSI sur FC . . . 111
Préparez-vous au provisionnement des volumes SMB . . . 114

Configuration et gestion des systèmes back-end . . . 115
Configuration des systèmes back-end . . . 115
Azure NetApp Files . . . 116
Google Cloud NetApp volumes . . . 135
Configurer un système NetApp HCI ou SolidFire backend . . . 168
Pilotes SAN de ONTAP . . . 173
Pilotes NAS ONTAP . . . 204
Amazon FSX pour NetApp ONTAP . . . 242
Création de systèmes back-end avec kubectl . . . 281
Gestion des systèmes back-end . . . 288
Créer et gérer des classes de stockage . . . 298
Créer une classe de stockage . . . 298
Gérer les classes de stockage . . . 301
Provisionnement et gestion des volumes . . . 303
Provisionner un volume . . . 303
Déplacer un volume . . . 307
Développement des volumes . . . 310
Comprendre les limites du sous-système RWX NVMe . . . 322
Évolutivité du contrôleur . . . 324
Extension automatique du volume . . . 329
Gérer les stratégies Autogrow . . . 335
Importer des volumes . . . 345
Personnaliser les noms et les étiquettes des volumes . . . 358
Partager un volume NFS entre les espaces de noms . . . 361
Cloner des volumes entre des espaces de noms . . . 365
Réplication de volumes à l'aide de SnapMirror . . . 368
Utiliser la topologie CSI . . . 374
Travailler avec des instantanés . . . 382
Travailler avec des instantanés de groupes de volumes . . . 390
Gérer et surveiller Trident . . . 395
Mettez à niveau Trident . . . 395
Mettez à niveau Trident . . . 395
Mise à niveau avec l'opérateur . . . 396
Mise à niveau avec tridentctl . . . 401
Gérez Trident à l'aide de tridentctl . . . 402
Commandes et indicateurs globaux . . . 402
Options et indicateurs de commande . . . 404
Prise en charge des plug-ins . . . 410
Surveillez Trident . . . 410
Présentation . . . 410
Étape 1 : définir une cible Prometheus . . . 410
Étape 2 : créer un ServiceMonitor Prometheus . . . 411
Étape 3 : interroger les mesures Trident avec PromQL . . . 413
En savoir plus sur la télémétrie Trident AutoSupport . . . 414
Désactivez les mesures Trident . . . 415

Désinstaller Trident . . . 415
Déterminez la méthode d'installation d'origine . . . 415
Désinstallez l'installation d'un opérateur Trident . . . 415
Désinstallez un `tridentctl` installation . . . 416
Trident pour Docker . . . 417
Conditions préalables au déploiement . . . 417
Vérifier les exigences . . . 417
Outils NVMe . . . 419
Outils FC . . . 420
Déployez Trident . . . 422
Méthode de plug-in géré Docker (version 1.13/17.03 et ultérieure) . . . 422
Méthode traditionnelle (version 1.12 ou antérieure) . . . 424
Démarrez Trident au démarrage du système . . . 425
Mise à niveau ou désinstallation de Trident . . . 426
Mise à niveau . . . 427
Désinstaller . . . 428
Utilisation de volumes . . . 428
Créer un volume . . . 428
Supprimer un volume . . . 429
Clonez un volume . . . 429
Accéder aux volumes créés en externe . . . 431
Options de volume spécifiques au conducteur . . . 431
Collecte des journaux . . . 436
Collecte des journaux pour le dépannage . . . 436
Conseils généraux de dépannage . . . 437
Gestion de plusieurs instances Trident . . . 437
Étapes du plug-in géré par Docker (version 1.13/17.03 ou ultérieure) . . . 437
Étapes pour les versions traditionnelles (version 1.12 ou antérieure) . . . 438
Options de configuration du stockage . . . 438
Options de configuration globale . . . 438
Configuration ONTAP . . . 439
Configuration logicielle Element . . . 448
Problèmes et limites connus . . . 450
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. . . . 450
Les noms de volumes doivent comporter au moins 2 caractères. . . . 451
Docker Swarm présente certains comportements qui empêchent Trident de le prendre en charge avec toutes les combinaisons de stockage et de pilotes. . . . 451
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. . . . 451
Meilleures pratiques et recommandations . . . 452
Déploiement . . . 452
Déploiement dans un namespace dédié . . . 452
Utilisez les quotas et les limites des plages pour contrôler la consommation du stockage . . . 452

Configuration de stockage sous-jacente . . . 452
Présentation de la plateforme . . . 452
Meilleures pratiques pour ONTAP et Cloud Volumes ONTAP . . . 452
Les meilleures pratiques pour SolidFire . . . 457
Où trouver plus d'informations ? . . . 459
Intégrez Trident . . . 459
Choix et déploiement du conducteur . . . 459
Conception de classe de stockage . . . 462
Conception de pool virtuel . . . 463
Opérations de volume . . . 464
Services de metrics . . . 468
Protection des données et reprise d'activité . . . 469
Réplication et restauration Trident . . . 469
Réplication et restauration des SVM . . . 470
Réplication et restauration de volume . . . 471
Protection des données Snapshot . . . 471
Automatisation du basculement des applications avec état grâce à Trident . . . 471
Détails sur le détachement forcé . . . 471
Détails sur le basculement automatique . . . 472
Sécurité . . . 477
Sécurité . . . 477
Configuration de clé unifiée Linux (LUKS) . . . 478
Chiffrement Kerberos à la volée . . . 485
Protégez vos applications avec Trident Protect . . . 493
Découvrez Trident Protect . . . 493
Et la suite ? . . . 493
Installer Trident Protect . . . 493
Exigences de Trident Protect . . . 493
Installez et configurez Trident Protect . . . 497
Installez le plugin CLI Trident Protect . . . 502
Personnaliser l'installation de Trident Protect . . . 506
Gérer Trident Protect . . . 511
Gérer l'autorisation et le contrôle d'accès de Trident Protect . . . 511
Surveiller les ressources de Trident Protect . . . 518
Générer un ensemble de support Trident Protect . . . 523
Amélioration de Trident Protect . . . 525
Gérez et protégez les applications . . . 527
Utilisez les objets Trident Protect AppVault pour gérer les compartiments . . . 527
Définissez une application de gestion avec Trident Protect . . . 541
Protégez les applications à l'aide de Trident Protect . . . 547
Restauration des applications . . . 559
Répliquez les applications à l'aide de NetApp SnapMirror et Trident Protect . . . 585
Migrer les applications à l'aide de Trident Protect . . . 603
Protégez les machines virtuelles KubeVirt avec Trident Protect . . . 607
Gérer les hooks d'exécution de Trident Protect . . . 619

Désinstallez Trident Protect . . . 631
Blogs de Trident et Trident Protect . . . 632
Blogs Trident . . . 632
Blogs de Trident Protect . . . 632
Connaissances et support . . . 634
Foire aux questions . . . 634
Questions générales . . . 634
Installez et utilisez Trident sur un cluster Kubernetes . . . 634
Dépannage et support . . . 635
Mettez à niveau Trident . . . 636
Gestion des systèmes back-end et des volumes . . . 637
Dépannage . . . 641
Dépannage général . . . 641
Échec du déploiement de Trident avec l'opérateur . . . 643
Échec du déploiement de Trident avec `tridentctl` . . . 645
Retirez complètement les Trident et les CRD . . . 645
Échec de l'annulation du transfert de nœud NVMe avec les espaces de noms de bloc bruts RWX o Kubernetes 1.26 . . . 646
Les clients NFSv4.2 signalent un « argument non valide » après la mise à niveau ONTAP alors qu'ils s'attendent à ce que « v4.2-xattrs » soit activé . . . 647
Assistance . . . 647
Cycle de vie du support Trident . . . 647
Auto-assistance . . . 648
Soutien de la communauté . . . 648
Support technique NetApp . . . 648
Pour en savoir plus . . . 648
Référence . . . 649
Ports Trident . . . 649
Présentation . . . 649
API REST Trident . . . 651
Quand utiliser l'API REST . . . 651
Avec l'API REST . . . 652
Options de ligne de commande . . . 652
Journalisation . . . 652
Kubernetes . . . 653
Docker . . . 653
REPOS . . . 653
Kubernetes et objets Trident . . . 654
Comment les objets interagissent-ils les uns avec les autres ? . . . 654
Kubernetes `PersistentVolumeClaim` objets . . . 654
Kubernetes `PersistentVolume` objets . . . 656
Kubernetes `StorageClass` objets . . . 656
Kubernetes `VolumeSnapshotClass` objets . . . 660
Kubernetes `VolumeSnapshot` objets . . . 661

Kubernetes `VolumeSnapshotContent` objets . . . 661
Objets Kubernetes `VolumeGroupSnapshotClass` . . . 661
Objets Kubernetes `VolumeGroupSnapshot` . . . 662
Objets Kubernetes `VolumeGroupSnapshotContent` . . . 662
Kubernetes `CustomResourceDefinition` objets . . . 663
ObjetsTrident `StorageClass` . . . 663
Objets back-end Trident . . . 663
ObjetsTrident `StoragePool` . . . 664
ObjetsTrident `Volume` . . . 664
ObjetsTrident `Snapshot` . . . 666
ObjetTrident `ResourceQuota` . . . 666
Normes de sécurité de pod (PSS) et contraintes de contexte de sécurité (SCC) . . . 667
Contexte de sécurité Kubernetes requis et champs associés . . . 668
Normes de sécurité du pod (PSS) . . . 668
Politiques de sécurité des pods (PSP) . . . 669
Contraintes de contexte de sécurité (SCC) . . . 670
Mentions légales . . . 673
Droits d’auteur . . . 673
Marques déposées . . . 673
Brevets . . . 673
Politique de confidentialité . . . 673
Source ouverte . . . 673

Documentation Trident 26.06

Notes de mise à jour

Quoi de neuf

Les notes de publication fournissent des informations sur les nouvelles fonctionnalités, les améliorations et les correctifs de bogues de la dernière version de NetApp 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.

Quoi de neuf dans la version 26.06

Découvrez les nouveautés de NetApp Trident et Trident Protect, y compris les améliorations, les correctifs et les suppressions.

Trident

Améliorations

- **Kubernetes** :
  - Ajout de la prise en charge de la disponibilité générale (GA) pour la concurrence des pilotes ONTAP-NAS-Economy et ONTAP-SAN-Economy, faisant passer cette fonctionnalité de la version d'évaluation technique à la version finale, et ajout de la prise en charge de la concurrence GA pour les pilotes Azure NetApp Files (ANF) et SolidFire.
  - Ajout de la prise en charge de Kubernetes 1.36.
  - Ajout de la prise en charge de la disponibilité générale (GA) pour l'extension automatique des volumes avec les pilotes NAS et SAN Google Cloud NetApp Volumes (GCNV).
  - Ajout du placement équilibré Unified ONTAP pour une sélection optimale des agrégats lors du provisionnement de nouveaux volumes ("Problème n° 64").
  - Ajout de la prise en charge ONTAP AFX pour le pilote ONTAP-NAS-Flexgroup dans ONTAP.
  - Ajout d'une intégration avec le NetApp Shift Toolkit pour simplifier les migrations de machines virtuelles lors de l'utilisation du Red Hat OpenShift Migration Toolkit for Virtualization.
  - Ajout de la prise en charge de Cloud Identity pour le pilote SAN GCNV.
  - Ajout de la prise en charge des volumes subordonnés pour la propagation de la publication de volumes Trident (TVP) et les flux de travail d'extension automatique.
  - Ajout de la prise en charge des mises à jour de volumes et des instantanés de groupes de volumes lors de l'exécution de Trident avec la concurrence activée.
  - Ajout d'une limitation dynamique du débit gRPC pour GCNV.
  - Ajout d'une sonde d'exécution de disponibilité du contrôleur et optimisation des paramètres QPS et Burst de l'API Kubernetes par défaut pour le contrôleur et les sidecars.
  - Amélioration du séquencement des opérations sur les nœuds grâce au découplage des étapes d'attachement, de configuration du chiffrement et de formatage du système de fichiers.
  - Les systèmes de fichiers racine des conteneurs Trident ont été montés en lecture seule pour améliorer la sécurité.

- Ajout d'une suppression au mieux des instantanés pour les instantanés de volumes clonés dans les backends ONTAP lorsque l'instantané source d'un volume cloné est supprimé.
- Ajout de la reconnaissance des PVC de clone de sauvegarde pour empêcher les PVC temporaires créés par Veeam Kasten, Trilio et Cohesity de rester bloqués dans la file d'attente de récupération de volume ONTAP.
- Gestion améliorée des ARN AWS pour la prise en charge de régions spécifiques.
- Empêcher le rétrécissement des Flexvols lors de l'opération de redimensionnement pour le pilote ONTAP-SAN-ECONOMY.

Améliorations expérimentales

> Ne pas utiliser dans des environnements de production.

- **[Aperçu technique]** : Ajout de la prise en charge de l'importation de qtree ONTAP-NAS-Economy.
- **[Aperçu technique]** : Ajout de la prise en charge de la concurrence pour les pilotes ASA-R2 (SAN et NVMe).
- **[Aperçu technique]** : Ajout du transport HTTP GCNV en mode ONTAP pour les pilotes ONTAP-SAN et ONTAP-NAS.

Correctifs

- **Kubernetes** :
  - Correction des conditions de concurrence et de la gestion d'état pour les opérations simultanées de publication/dépublication, de clonage, de mise en cache et de mise à jour du backend.
  - Correction des conditions de concurrence des règles d'export et de la gestion simultanée de la publication/dépublication pour les volumes subordonnés et les clones en lecture seule dans les pilotes ONTAP-NAS et ONTAP-NAS-Economy.
  - Correction de la découverte ONTAP-NAS-Economy FlexVol après un basculement MetroCluster lorsque les règles de snapshot diffèrent ("Problème n° 1082").
  - Correction des tentatives de suppression de volume ONTAP-NAS-Economy qui interrompaient les suppressions qtree de longue durée ("Problème n° 1121").
  - Correction du comportement d'importation et de redimensionnement d'ONTAP-SAN et d'ONTAP-SAN-Economy, y compris la validation de fsType, la gestion des métadonnées de volume et le comportement du mode d'extension automatique.
  - Correction d'une condition de concurrence d'espace de noms NVMe/TCP lors de la création simultanée de volumes ("Problème n° 1089").
  - Correction des échecs de redimensionnement iSCSI qui survenaient lorsque les chemins et les portails disponibles ne correspondaient pas, et amélioration du comportement lors des redimensionnements répétés.
  - Correction des échecs de suppression de volumes ONTAP-SAN dus à des conditions d'occupation des volumes.
  - Correction de la gestion des erreurs ONTAP REST et amélioration du comportement du délai d'expiration de l'état SVM.
  - Correction de la gestion des pools zonaux SAN GCNV et des groupes d'hôtes, normalisation de la conversion des identifiants de volume et normalisation des requêtes de création inférieures à la taille minimale.

- Correction de la sélection du pool de capacité natif GCNV afin d'exclure les pools en mode ONTAP.
- Correction du basculement KubeVirt pour les machines virtuelles utilisant Trident Automated Workload Failover afin d'accélérer le basculement des machines virtuelles.
- Correction des problèmes de contrôle d'accès basé sur les rôles (RBAC) liés à l'installation gérée par l'opérateur, notamment les nœuds ClusterRoles manquants et la réconciliation incorrecte de ClusterRole et ClusterRoleBinding lors des mises à niveau. Cela s'applique aux installations basées sur l'opérateur et sur Helm qui utilisent le chart Trident Operator.
- Correction des différences persistantes GitOps sur le TridentConfigurator et les CRD associés en omettant les valeurs de priorité par défaut et en corrigeant l'indentation du modèle YAML ("Problème n° 915", "Problème n° 1120").
- Correction du patching des CRD TridentVolumePublication lors d'une mise à niveau Helm ("Problème n° 1151").
- Correction de l'ordre de démarrage CSI en démarrant le service CSI gRPC avant les nouvelles tentatives d'enregistrement des nœuds.
- Correction des superpositions de configuration de clonage SMB sécurisé lors d'opérations simultanées.
- Correction des annotations manquantes sur le pod du contrôleur Trident après une mise à niveau ("Problème n°1004").
- Amélioration du provisionnement lent des qtrees sur les grands SVM ("Problème n° 1142").
- Correction apportée à ControllerPublish pour utiliser le type de système de fichiers de configuration du volume au lieu d'un type de système de fichiers par défaut incorrect.
- Correction d'un problème où les périphériques dm orphelins pouvaient provoquer des échecs CSI répétés NodeUnstageVolume.
- Amélioration du nettoyage des nœuds obsolètes et des chemins multiples grâce à la suppression des artefacts de chemins multiples fantômes et des ressources TridentNode obsolètes après le nettoyage des nœuds supprimés de manière logique.

Protection Trident

Améliorations

- Ajout de l'intégration de l'opérateur OpenShift Service CA pour automatiser la gestion des certificats TLS pour le webhook d'admission Trident Protect. Activez-la en définissant `useOpenshiftServiceCA` sur `true` dans vos valeurs Helm. Pour en savoir plus, consultez "Personnaliser l'installation de Trident Protect".
- Ajout de la prise en charge de la sauvegarde et de la restauration des machines virtuelles KubeVirt sur OpenShift Virtualization. Protégez toutes les machines virtuelles d'un espace de noms ou ciblez des machines virtuelles spécifiques par leur nom, et restaurez sélectivement des machines virtuelles individuelles à partir d'une sauvegarde ou d'un instantané. Pour en savoir plus, consultez "Protégez les machines virtuelles KubeVirt avec Trident Protect".
- Prise en charge des transformations de ressources ajoutée, vous permettant d'ajouter, de supprimer et de modifier les attributs des ressources Kubernetes lors du processus de restauration. Pour en savoir plus, consultez "Modifiez les ressources au fur et à mesure de leur restauration à l'aide de ressources personnalisées".
- Ajout de la prise en charge de `storageClassMapping` dans AppMirrorRelationship (AMR), permettant le mappage des classes de stockage pour les PVC individuels au sein d'une même définition AMR. Pour en savoir plus, consultez "Répliquez les applications à l'aide de NetApp SnapMirror et Trident Protect.".

Correctifs

- Correction d'un problème de restauration de machines virtuelles inter-clusters qui échouait avec une erreur de webhook d'admission lorsque l'espace de noms de l'application source n'existait pas sur le cluster de destination.
- Correction d'un problème de restauration sélective sur place de machines virtuelles qui échouait lors de la création de l'application en raison d'une incompatibilité de spécifications avec une ressource personnalisée d'application existante ; l'application existante est désormais réutilisée.
- Correction d'un problème où les tâches de restauration sur place de machines virtuelles restaient bloquées dans un état `Running` lorsque la machine virtuelle et son PersistentVolumeClaim étaient supprimés ; un PersistentVolumeClaim manquant est désormais considéré comme une erreur fatale.
- Correction d'un problème qui déclenchait incorrectement, lors de la modification d'une planification de protection existante, une sauvegarde ou une exécution de snapshot immédiate.
- Correction du problème où le finaliseur d'instantané tentait de récupérer la ressource personnalisée de l'application après sa suppression ; le finaliseur ignore désormais l'opération lorsque l'application est introuvable.
- La capacité de stockage protégée fixe augmente considérablement après une restauration sur place d'une machine virtuelle grâce à l'utilisation d'une copie partielle lors de l'étape de copie PersistentVolumeClaim.
- Correction d'un problème où Trident Protect supprimait l'objet snapshot Kubernetes avant de confirmer la suppression du snapshot sous-jacent dans ONTAP ; le finaliseur attend désormais la `VolumeSnapshot`suppression.
- Correction d'un problème où les instantanés ne se poursuivaient pas lorsqu'une opération de gel de machine virtuelle échouait ; les opérations de gel et de dégel se poursuivent désormais en cas d'échec.
- Correction d'un problème où les applications de machines virtuelles KubeVirt recréaient incorrectement les PersistentVolumeClaims lors des opérations d'inversion, de resynchronisation inverse et de resynchronisation AppMirrorRelationship.

Nouveautés de la version 26.02.1

Découvrez les nouveautés de NetApp Trident et Trident Protect, y compris les améliorations, les correctifs et les suppressions.

Trident

Correctifs

- **Kubernetes** :
- Correction du comportement SAN de Google Cloud NetApp Volumes (GCNV) pour la prise en charge des pools zonaux Flex Unified pour les charges de travail de stockage bloc (iSCSI). Les configurations zonales et régionales sont désormais prises en charge.
- Correction de la gestion des pools NAS UNIFIED Google Cloud NetApp Volumes (GCNV) en améliorant la création, la validation, la dénomination et la recherche de volumes de secours lorsque `InternalID` est absent.
- Correction d'un problème où les opérations de redimensionnement de périphériques iSCSI pouvaient échouer lorsque les chemins et portails disponibles ne correspondaient pas.
- Correction d'un problème intermittent où `blkid` identifiait incorrectement le type de système de fichiers pour les périphériques LUKS, empêchant la création de nouveaux volumes.
- Correction d'un problème où les opérations de publication et de préparation LUKS à grande échelle

pouvaient bloquer le contrôleur Trident. Les utilisateurs doivent désormais conserver les phrases de passe LUKS associées aux volumes.

- Correction d'un problème où les partitions iSCSI multipath pouvaient apparaître comme des périphériques fantômes, empêchant les opérations CSI `NodeUnstage` de réussir.
- Correction d'une condition de concurrence d'espace de noms NVMe/TCP lors de la création simultanée de volumes (Issue #1089).
- Correction des opérations de clonage d'instantanés simultanées lorsque les instantanés étaient exécutés dans la même seconde.
- Correction d'un problème empêchant les machines virtuelles KubeVirt utilisant `dataVolumes` de basculer avec Trident Automated Workload Failover.
- Correction d'une erreur qui se produisait lors de la suppression d'une ressource personnalisée non valide `TridentMirrorRelationship` lors de l'utilisation du noyau concurrent de Trident.
- Correction de l'indentation dans la `tridentactionmirrorupdates`définition YAML du CRD Kubernetes (Issue #1120).
- Correction de l'appel API de suppression asynchrone de qtree REST (Issue #1121).
- Correction apportée `ControllerPublish`pour utiliser le type de système de fichiers de configuration du volume au lieu d'un type de système de fichiers par défaut incorrect.
- Correction de la journalisation au démarrage du contrôleur pour signaler correctement l'état des pilotes simultanés.
- Correction de la gestion des backends suspendus dans Trident concurrent.
- Gestion améliorée des ARN AWS pour la prise en charge de régions spécifiques.

Protection Trident

Correctifs

Mise à jour des dépendances de Trident Protect pour corriger les failles de sécurité :

- gRPC-Go : "CVE-2026-33186"
- Go x/crypto SSH :"CVE-2024-45337"
- OpenTelemetry-Go : "CVE-2026-24051"
- Exécution Go : "CVE-2025-22871"
- Dépendances supplémentaires :"CVE-2025-23109"

Quoi de neuf dans la version 26.02

Découvrez les nouveautés de NetApp Trident et Trident Protect, y compris les améliorations, les correctifs et les suppressions.

Trident

Améliorations

- **Kubernetes** :
  - Ajout de la prise en charge en disponibilité générale (GA) de la concurrence pour ONTAP-NAS (NFS uniquement), ONTAP-SAN (iSCSI, FCP, NVMe) et Google Cloud NetApp Volumes (GCNV), faisant passer cette fonctionnalité de Tech Preview à la version finale. Voir la "Documentation sur la scalabilité

du contrôleur" pour plus d'informations.

- Ajout de la prise en charge de l'extension automatique des volumes dans Trident basée sur des stratégies AutoGrow personnalisées et définies par l'utilisateur dans Trident. Voir le "Documentation sur l'extension automatique du volume" pour plus d'informations.
- Amélioration de la concurrence des nœuds Trident pour optimiser l'évolutivité des opérations sur les volumes NVMe. Voir la "Documentation sur les volumes NVMe" pour plus d'informations.
- Ajout de la prise en charge du hiérarchisation automatique des volumes NAS GCNV via les annotations PVC (`tieringPolicy`, `tieringMinimumCoolingDays`), y compris la sélection des pools et l'héritage des clones. Voir la "Configurez la hiérarchisation automatique pour la documentation Google Cloud NetApp Volumes" pour plus d'informations.
- Ajout de la prise en charge des volumes SAN bloc GCNV (iSCSI) utilisant le `google-cloud-netapp-volumes-san` driver, y compris le provisionnement, le mappage des groupes d'hôtes par nœud pour l'accès aux LUN et les opérations de clonage à partir de volumes. Voir le "Documentation sur la configuration des blocs Google Cloud NetApp Volumes" pour plus d'informations.
- Ajout de la prise en charge de la configuration automatique du backend pour Amazon FSx for NetApp ONTAP. Lorsque vous créez un StorageClass qui inclut les paramètres requis, Trident crée automatiquement le backend correspondant et la VolumeSnapshotClass, si nécessaire. Voir le "Documentation de configuration du backend Amazon FSx for NetApp ONTAP" pour plus d'informations.
- Ajout de la prise en charge de différents clouds Microsoft Azure, tels qu'Azure Government et Azure China, et d'une configuration cloud personnalisée pour Azure NetApp Files backends. Voir le "Documentation de configuration du backend Azure NetApp Files" pour plus d'informations.
- Ajout de la prise en charge de Kubernetes 1.35. Voir le "Documentation des exigences" pour plus d'informations.

Améliorations expérimentales

Ne pas utiliser dans des environnements de production.

- **[Aperçu technique]** : Ajout de la prise en charge de la concurrence pour les pilotes ONTAP-NAS-Economy et ONTAP-SAN-Economy.

Problèmes connus

Il existe un problème connu avec le `csi-snapshotter` sidecar. Dans toutes les versions de Kubernetes, VolumeGroupSnapshots `v1beta1` empêche VolumeSnapshots d'atteindre l' `ReadyToUse` état.

Deux solutions de contournement sont disponibles :

1. Supprimez les CRD VolumeGroupSnapshots pour désactiver VolumeGroupSnapshots, puis réinstallez Trident.
2. Installez VolumeGroupSnapshots `v1beta2` et `snapshot-controller` la version 8.4.0 ou ultérieure, puis réinstallez Trident. VolumeGroupSnapshots ne fonctionnent pas sur les versions de Kubernetes antérieures à la v1.34.

Correctifs

- **Kubernetes** :

- Correction d'un problème où la dépublication d'un clone en lecture seule supprimait les règles d'export du volume source dans les pilotes ONTAP-NAS, ONTAP-NAS-Economy et Google Cloud NetApp Volumes "Problème n°1086".
- Les images kubectl ont été remplacées par des variantes légères basées sur Alpine afin de prévenir les échecs de récupération suite à la dépréciation des images publiques de Bitnami "Problème n°1080".
- Correction de la préservation des annotations du déploiement existant lors de la mise à niveau de Trident "Problème n°1004".
- Autoriser le clonage entre différentes classes de stockage si les deux classes de stockage pointent vers le même backend "Problème n°1104".
- Correction des échecs de préparation des nœuds dus aux délais d'attente dans les environnements cloud présentant des latences réseau. Augmentation des valeurs de délai d'attente pour les installations cloud.
- Correction d'un problème lors de la création de LUN qui empêchait la définition de l'attribut de type de système de fichiers lorsque le processus entrait dans un état de nouvelle tentative.
- Correction de la recherche de volume via l'API REST afin d'ignorer l'état du volume, empêchant les faux négatifs lors des requêtes de volume.
- Amélioration de l'efficacité du contrôleur Trident pour le pilote ontap-nas-economy lorsqu'il est utilisé à grande échelle.
- Définir internalID lors de l'importation du LUN dans le pilote ontap-san-economy.
- Augmentation des limites de requêtes Azure Resource Graph pour gérer davantage de sous-réseaux.
- Amélioration des délais d'expiration de la division des clones CSI et ONTAP pour éviter les conditions de concurrence avec certaines applications de sauvegarde "Problème #1098", "Problème n°1100".
- Correction de la suppression des messages d'erreur LUKS "Problème n°1069".
- Correction de la gestion des mappeurs LUKS obsolètes pour les protocoles iSCSI et NVMe. La logique de nettoyage améliorée empêche les échecs de montage dus aux mappeurs de périphériques orphelins.

- Limites d'échelle corrigées pour les volumes RWX NVMe.
- Mise à jour du package OpenTelemetry-Go pour corriger "CVE-2026-24051".

Protection Trident

Améliorations

- Trident Protect désactive désormais automatiquement les planifications de protection et annule les opérations en cours avant une restauration sur place, puis les réactive une fois la restauration terminée. Pour en savoir plus, consultez "Restaurez les applications à l'aide de Trident Protect".
- Ajout du champ `runImmediately` au CR de planification et du paramètre CLI `--run-immediately` pour déclencher une sauvegarde ou un instantané immédiat lors de la création de la planification. Pour en savoir plus, consultez "Créez un calendrier de protection des données".
- Ajout de la prise en charge de la spécification d'un nom personnalisé pour l'application restaurée à l'aide du `destinationApplicationName` champ dans le CR de restauration ou du `--destination-app -name` paramètre CLI. Pour en savoir plus, consultez "Restaurez les applications à l'aide de Trident Protect".

Correctifs

- Correction des échecs de restauration causés par la création de pods avant que leurs comptes de service requis ne soient disponibles.
- Correction de `Roles` et `RoleBindings` qui étaient ignorés lors des restaurations d'applications.
- Correction du problème où le nom du cluster d'origine ne s'affichait pas dans `tridentctl-protect get appvaultcontent` la sortie, malgré une configuration correcte.
- Correction des erreurs de restauration de Kopia qui étaient ignorées en raison du manque de `pipefail` prise en charge.
- Correction des échecs de snapshots et de sauvegardes causés par des filtres de ressources qui excluaient les volumes persistants.
- Correction d'une restauration incorrecte de PVC dans les applications multi-espaces de noms avec des PVC portant le même nom dans différents espaces de noms, ce qui pouvait entraîner une perte de données.

Quoi de neuf dans la version 25.10

Découvrez les nouveautés de Trident et Trident Protect, notamment les améliorations, les correctifs et les fonctionnalités obsolètes.

Trident

Améliorations

- **Kubernetes** :
  - Ajout de la prise en charge des instantanés de groupes de volumes CSI avec les API Kubernetes d'instantanés de groupes de volumes v1beta1 pour les pilotes ONTAP-NAS NFS et ONTAP-SAN-Economy, en plus de ONTAP-SAN (iSCSI et FC). Voir"Travailler avec des instantanés de groupes de volumes" .
  - Ajout de la prise en charge du basculement automatique de la charge de travail avec détachement forcé du volume pour les pilotes ONTAP-NAS et ONTAP-NAS-Economy (à l'exclusion de SMB dans les deux pilotes NAS), ainsi que pour les pilotes ONTAP-SAN et ONTAP-SAN-Economy. Voir"Automatisation du basculement des applications avec état grâce à Trident" .
  - Amélioration de la concurrence des nœuds Trident pour une meilleure évolutivité des opérations sur les nœuds pour les volumes FCP.
  - Ajout de la prise en charge ONTAP AFX pour le pilote ONTAP NAS. Voir"Options et exemples de configuration du NAS ONTAP" .
  - Ajout de la prise en charge de la configuration des demandes et des limites de ressources CPU et mémoire pour les conteneurs Trident via les valeurs CR et Helm du graphique TridentOrchestrator. ("Numéro 1000" ,"Numéro 927" ,"Numéro 853" ,"Numéro 592" ,"Numéro 110").
  - Ajout de la prise en charge FC pour la personnalité ASAr2. Voir"Options et exemples de configuration des SAN ONTAP" .
  - Ajout d'une option permettant de diffuser les métriques Prometheus via HTTPS au lieu de HTTP. Voir"Surveillez Trident" .
  - Ajout d'une option `--no-rename` lors de l'importation d'un volume pour conserver le nom d'origine mais laisser Trident gérer le cycle de vie du volume. Voir"Importer des volumes" .
  - Le déploiement de Trident s'exécute désormais au niveau de priorité « system-cluster-critical ».

- Ajout d'une option permettant au contrôleur Trident d'utiliser le réseau hôte via Helm, l'opérateur et Tridentctl ("Numéro 858").
- Ajout de la prise en charge manuelle de la QoS au pilote ANF, la rendant prête pour la production dans Trident 25.10 ; cette amélioration expérimentale a été introduite dans Trident 25.06.

Améliorations expérimentales

> Ne pas utiliser dans des environnements de production.

- **[Aperçu technique]** : Ajout de la prise en charge de la concurrence pour ONTAP-NAS (NFS uniquement) et ONTAP-SAN (NVMe pour ONTAP 9 unifié), en plus de l'aperçu technique existant pour le pilote ONTAP-SAN (protocoles iSCSI et FCP dans ONTAP 9 unifié).

Correctifs

- **Kubernetes** :
  - Correction de l'incohérence du nom du conteneur CSI node-driver-registrar en normalisant Linux DaemonSet en node-driver-registrar pour correspondre à la dénomination des images de conteneurs et de DaemonSet Windows.
  - Correction d'un problème où les politiques d'exportation pour les qtrees hérités n'étaient pas correctement mises à jour.
- **Openshift** :
  - Correction d'un problème de démarrage du pod de nœud Trident sur les nœuds Windows dans Openshift dû au fait que SCC avait allowHostDirVolumePlugin défini sur false ("Numéro 950").
- Correction du problème de configuration du QPS de l'API Kubernetes via Helm ("Numéro 975").
- Correction d'un problème d'impossibilité de monter une revendication de volume persistant (PVC) basée sur un instantané d'un système de fichiers XFS basé sur NVMe PVC sur le même nœud Kubernetes.
- Correction d'un problème de changement d'UUID après le redémarrage de l'hôte/Docker en mode NDVP en ajoutant des noms de sous-système uniques/partagés par backend (par exemple, netappdvp_subsystem).
- Correction des erreurs de montage pour les volumes iSCSI lors de la mise à niveau de Trident des versions antérieures à 23.10 vers la version 24.10 et supérieures, résolution du problème « SANType invalide ».
- Correction d'un problème où l'état du backend Trident ne passait pas à l'état en ligne/hors ligne sans redémarrage du contrôleur Trident .
- Correction d'un problème de concurrence intermittent provoquant un redimensionnement lent des PVC.
- Correction d'un problème où les instantanés n'étaient pas supprimés en cas d'échec de clonage de volume.
- Correction d'un problème empêchant la désactivation du volume lorsque son chemin d'accès au périphérique était modifié par le noyau.
- Correction d'un problème de désinstallation du volume dû à un périphérique LUKS déjà fermé.
- Correction d'un problème où des opérations de stockage lentes entraînaient des erreurs ContextDeadline.
- L'opérateur Trident attendra un délai d'expiration k8s configurable pour vérifier la version de Trident .

Protection Trident

NetApp Trident Protect offre des fonctionnalités avancées de gestion des données d'application qui améliorent la fonctionnalité et la disponibilité des applications Kubernetes avec état, prises en charge par les systèmes de stockage NetApp ONTAP et le provisionneur de stockage NetApp Trident CSI.

Améliorations

- Ajout d'annotations pour contrôler les délais d'expiration des CR d'instantané pour les CR planifiées et de sauvegarde :
  - `protect.trident.netapp.io/snapshot-completion-timeout`
  - `protect.trident.netapp.io/volume-snapshots-ready-to-use-timeout`
  - `protect.trident.netapp.io/volume-snapshots-created-timeout`

  Voir"Prise en charge des annotations de sauvegarde et de planification" .
- Ajout d'une annotation à la CR planifiée pour configurer le délai d'expiration de la liaison PVC, qui sera utilisé par la CR de sauvegarde : `protect.trident.netapp.io/pvc-bind-timeout-sec` .
  Voir"Prise en charge des annotations de sauvegarde et de planification" .
- Amélioré `tridentctl-protect` Listes de sauvegardes et d'instantanés avec un nouveau champ pour indiquer les échecs des hooks d'exécution.

Modifications apportées le 25/06/2020

Trident

Correctifs

- **Kubernetes** : problème critique résolu où des périphériques iSCSI incorrects étaient découverts lors du détachement de volumes des nœuds Kubernetes.

Modifications dans la version 25.06.1

Trident

> Pour les clients utilisant SolidFire, veuillez ne pas effectuer la mise à jour vers la version 25.06.1 en raison d'un problème connu lors de la dépublication des volumes. La version 25.06.2 sera bientôt disponible pour corriger ce problème.

Correctifs

- **Kubernetes** :
  - Correction d'un problème où les NQN n'étaient pas vérifiés avant d'être démappés des sous-systèmes.
  - Correction d'un problème où plusieurs tentatives de fermeture d'un périphérique LUKS entraînaient des échecs lors du détachement des volumes.
  - Correction du volume iSCSI non mis en scène lorsque le chemin du périphérique a changé depuis sa création.
  - Clonage de blocs de volumes sur plusieurs classes de stockage.
- **OpenShift** : un problème a été résolu où la préparation du nœud iSCSI échouait avec OCP 4.19.

- Augmentation du délai d'expiration lors du clonage d'un volume à l'aide des backends SolidFire ("Question no 1008").

Modifications dans la version 25.06

Trident

Améliorations

- **Kubernetes** :
  - Ajout de la prise en charge des instantanés de groupe de volumes CSI avec `v1beta1` API Kubernetes d'instantané de groupe de volumes pour le pilote iSCSI ONTAP-SAN. Voir "Travailler avec des instantanés de groupes de volumes".

    > VolumeGroupSnapshot est une fonctionnalité bêta de Kubernetes avec des API bêta. Kubernetes 1.32 est la version minimale requise pour VolumeGroupSnapshot.

  - Ajout de la prise en charge d' ONTAP ASA r2 pour NVMe/TCP en plus d'iSCSI. Voir"Options et exemples de configuration des SAN ONTAP" .
  - Prise en charge SMB sécurisée ajoutée pour les volumes ONTAP-NAS et ONTAP-NAS-Economy. Les utilisateurs et groupes Active Directory peuvent désormais être utilisés avec les volumes SMB pour une sécurité renforcée. Voir "Activer le SMB sécurisé".
  - Concurrence de nœud Trident améliorée pour une plus grande évolutivité des opérations de nœud pour les volumes iSCSI.
  - Ajouté `--allow-discards` lors de l'ouverture des volumes LUKS pour autoriser les commandes discard/TRIM pour la récupération d'espace.
  - Performances améliorées lors du formatage des volumes chiffrés LUKS.
  - Nettoyage LUKS amélioré pour les périphériques LUKS défaillants mais partiellement formatés.
  - Idempotence de nœud Trident améliorée pour la connexion et le détachement de volumes NVMe.
  - Ajouté `internalID` champ à la configuration du volume Trident pour le pilote ONTAP-SAN-Economy.
  - Ajout de la prise en charge de la réplication de volume avec SnapMirror pour les backends NVMe. Voir "Réplication de volumes à l'aide de SnapMirror".

Améliorations expérimentales

> Ne pas utiliser dans des environnements de production.

- [Aperçu technique] Opérations simultanées du contrôleur Trident activées via le `--enable -concurrency` Indicateur de fonctionnalité. Cela permet aux opérations du contrôleur de s'exécuter en parallèle, améliorant ainsi les performances dans les environnements très fréquentés ou de grande taille.

  > Cette fonctionnalité est expérimentale et prend actuellement en charge des flux de travail parallèles limités avec le pilote ONTAP-SAN (protocoles iSCSI et FCP).

- [Aperçu technique] Ajout de la prise en charge manuelle du QOS avec le pilote ANF.

Correctifs

- **Kubernetes** :
  - Correction d'un problème avec CSI NodeExpandVolume où les périphériques multi-chemins pouvaient être laissés avec des tailles incongrues lorsque les disques SCSI sous-jacents n'étaient pas disponibles.
  - Correction de l'échec de nettoyage des politiques d'exportation en double pour les pilotes ONTAP-NAS et ONTAP-NAS-Economy.
  - Correction des volumes GCNV qui utilisaient par défaut NFSv3 lorsque `nfsMountOptions` n'est pas défini ; désormais, les protocoles NFSv3 et NFSv4 sont pris en charge. Si `nfsMountOptions` n'est pas fourni, la version NFS par défaut de l'hôte (NFSv3 ou NFSv4) est utilisée.
  - Problème de déploiement résolu lors de l'installation de Trident à l'aide de Kustomize ("Question no 831").
  - Correction des politiques d'exportation manquantes pour les PVC créés à partir d'instantanés ("Question no 1016").
  - Problème résolu où les tailles de volume ANF ne sont pas automatiquement alignées sur des incréments de 1 Gio.
  - Problème résolu lors de l'utilisation de NFSv3 avec Bottlerocket.
- Problème résolu avec les volumes ONTAP-NAS-Economy s'étendant jusqu'à 300 To malgré les échecs de redimensionnement.
- Problème résolu où les opérations de division de clone étaient effectuées de manière synchrone lors de l'utilisation de l'API REST ONTAP.

Obsolescences :

- **Kubernetes** : mise à jour de la version minimale prise en charge de Kubernetes vers la version 1.27.

Protection Trident

NetApp Trident Protect offre des fonctionnalités avancées de gestion des données d'application qui améliorent la fonctionnalité et la disponibilité des applications Kubernetes avec état, prises en charge par les systèmes de stockage NetApp ONTAP et le provisionneur de stockage NetApp Trident CSI.

Améliorations

- Temps de restauration améliorés, offrant la possibilité d'effectuer des sauvegardes complètes plus fréquentes.
- Granularité améliorée de la définition de l'application et restauration sélective avec filtrage Groupe-Version-Type (GVK).
- Resynchronisation et réplication inverse efficaces lors de l'utilisation d'AppMirrorRelationship (AMR) avec NetApp SnapMirror, pour éviter la réplication PVC complète.
- Ajout de la possibilité d'utiliser l'identité du pod EKS pour créer des compartiments AppVault, supprimant ainsi la nécessité de spécifier un secret avec les informations d'identification du compartiment pour les clusters EKS.
- Ajout de la possibilité d'ignorer la restauration des étiquettes et des annotations dans l'espace de noms de restauration, si nécessaire.
- AppMirrorRelationship (AMR) vérifiera désormais l'extension du PVC source et effectuera l'extension appropriée sur le PVC de destination si nécessaire.

Correctifs

- Correction d'un bug où les valeurs d'annotation des instantanés précédents étaient appliquées aux instantanés plus récents. Toutes les annotations d'instantanés sont désormais correctement appliquées.
- Un secret a été défini pour le chiffrement du Data Mover (Kopia / Restic) par défaut, s'il n'est pas défini.
- Ajout de messages de validation et d'erreur améliorés pour la création d'appvault S3.
- AppMirrorRelationship (AMR) réplique désormais uniquement les PV dans l'état lié, pour éviter les tentatives infructueuses.
- Problème résolu où des erreurs s'affichaient lors de l'obtention d'AppVaultContent sur un AppVault avec un grand nombre de sauvegardes.
- Les instantanés VMSnapshots KubeVirt sont exclus des opérations de restauration et de basculement pour éviter les échecs.
- Problème résolu avec Kopia où les instantanés étaient supprimés prématurément en raison du calendrier de conservation par défaut de Kopia remplaçant ce qui avait été défini par l'utilisateur dans le calendrier.

Changements en 25.02.1

Trident

Correctifs

- **Kubernetes** :
  - Correction d'un problème dans l'opérateur Trident où les noms et les versions des images sidecar étaient mal renseignés lors de l'utilisation d'un registre d'images non par défaut ("Question no 983").
  - Résolution du problème d'échec de restauration des sessions multivoies lors d'un retour de basculement ONTAP ("Question no 961").

Changements en 25.02

À partir de Trident 25.02, le résumé des nouveautés fournit des détails sur les améliorations, les correctifs et les dépréciations pour les versions de Trident et de Trident Protect.

Trident

Améliorations

- **Kubernetes** :
  - Prise en charge de ONTAP ASA r2 pour iSCSI ajoutée.
  - Ajout de la prise en charge du détachement forcé pour les volumes ONTAP-NAS pendant les scénarios d'arrêt de nœud non normal. Les nouveaux volumes ONTAP-NAS utiliseront désormais des règles d'exportation par volume gérées par Trident. Chemin de mise à niveau fourni pour les volumes existants afin qu'ils puissent passer au nouveau modèle de règle d'export lors de l'annulation de la publication sans affecter les workloads actifs.
  - Ajout d'une annotation cloneFromSnapshot.
  - Prise en charge supplémentaire du clonage de volumes inter-espaces de noms.
  - L'analyse d'auto-rétablissement iSCSI améliorée corrige les problèmes pour lancer des renumérisations par hôte, canal, cible et ID de LUN exacts.

    - Prise en charge de Kubernetes 1.32.
- **OpenShift** :
    - Ajout de la prise en charge de la préparation automatique des nœuds iSCSI pour RHCOS sur les clusters ROSA.
    - Ajout de la prise en charge de la virtualisation OpenShift pour les pilotes ONTAP.
- Ajout de la prise en charge Fibre Channel sur le pilote ONTAP-SAN.
- Ajout de la prise en charge de NVMe LUKS.
- Passe à l'image de travail pour toutes les images de base.
- Ajout de la détection et de la journalisation de l'état de la connexion iSCSI lorsque les sessions iSCSI doivent être connectées mais ne sont pas ("Question no 961").
- Ajout de la prise en charge des volumes SMB avec le pilote google-cloud-NetApp-volumes
- Prise en charge supplémentaire permettant aux volumes ONTAP d'ignorer la file d'attente de restauration lors de la suppression.
- Ajout d'une prise en charge pour remplacer les images par défaut à l'aide de Shas au lieu de balises.
- Ajout de l'indicateur image-pull-secrets au programme d'installation de tridentctl.

Correctifs

- **Kubernetes** :
    - Correction des adresses IP de nœud manquantes des règles d'exportation automatique ("Question no 965").
    - Correction des règles d'exportation automatique pour passer prématurément à une règle de volume pour ONTAP-NAS-Economy.
    - Les informations d'identification de configuration du back-end ont été corrigées pour prendre en charge toutes les partitions ARN AWS disponibles ("Question no 913").
    - Ajout d'une option pour désactiver la réconciliation automatique du configurateur dans l'opérateur Trident ("Question no 924").
    - Ajout de securityContext pour conteneur csi-resizer ("Question no 976").

Protection Trident

NetApp Trident Protect offre des fonctionnalités avancées de gestion des données d'application qui améliorent la fonctionnalité et la disponibilité des applications Kubernetes avec état, prises en charge par les systèmes de stockage NetApp ONTAP et le provisionneur de stockage NetApp Trident CSI.

Améliorations

- Ajout de la prise en charge de la sauvegarde et de la restauration pour les machines virtuelles KubeVirt / OpenShift Virtualization pour les modes de stockage volumeMode : File et volumeMode : Block (périphérique brut). Cette prise en charge est compatible avec tous les pilotes Trident et améliore les fonctionnalités de protection existantes lors de la réplication du stockage à l'aide de NetApp SnapMirror avec Trident Protect.
- Ajout de la fonctionnalité de contrôle du comportement de gel au niveau des applications pour les environnements Kubevirt.
- Ajout de la prise en charge de la configuration des connexions proxy AutoSupport.

- Ajout de la capacité à définir un secret pour le cryptage des Data Movers (Kopia/Restic).
- Ajout de la possibilité d'exécuter manuellement un crochet d'exécution.
- Ajout de la possibilité de configurer les contraintes de contexte de sécurité (SCC) lors de l'installation de Trident Protect.
- Ajout de la prise en charge de la configuration de nodeSelector lors de l'installation de Trident Protect.
- Ajout de la prise en charge du proxy de sortie HTTP/HTTPS pour les objets AppVault.
- Filtre de ressources étendu pour permettre l'exclusion des ressources du cluster-scoped.
- Ajout de la prise en charge du jeton de session AWS dans les informations d'identification d'AppVault S3.
- Ajout de la prise en charge de la collecte de ressources après les crochets d'exécution pré-snapshot.

Correctifs

- Amélioration de la gestion des volumes temporaires pour ignorer la file d'attente de restauration du volume ONTAP.
- Les annotations SCC sont désormais restaurées aux valeurs d'origine.
- Amélioration de l'efficacité des restaurations grâce à la prise en charge des opérations parallèles.
- Prise en charge améliorée des délais d'expiration du crochet d'exécution pour les applications de plus grande taille.

Changements en 24.10.1

Améliorations

- **Kubernetes** : prise en charge supplémentaire de Kubernetes 1.32.
- Ajout de la détection et de la journalisation de l'état de la connexion iSCSI lorsque les sessions iSCSI doivent être connectées mais ne sont pas ("Question no 961").

Correctifs

- Correction des adresses IP de nœud manquantes des règles d'exportation automatique ("Question no 965").
- Correction des règles d'exportation automatique pour passer prématurément à une règle de volume pour ONTAP-NAS-Economy.
- Mise à jour des dépendances Trident et Trident-ASUP pour corriger CVE-2024-45337 et CVE-2024-45310.
- Déconnexion supprimée des portails non CHAP défectueux par intermittence pendant l'auto-rétablissement iSCSI ("Question no 961").

Changements en 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 18 Mio de surcharge aux LUN LUKS pour garantir que la taille du 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

  "Apprenez-en davantage sur Trident Protect".
- Ajout d'un nouveau drapeau `--k8s-api-qps` aux installateurs de 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.

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 ONTAPI (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 les tailles de volume inférieures à 20 Mio ("Problème[#885").
- Correction de la Trident pour empêcher la réduction des volumes FlexVol pendant le redimensionnement du pilote ONTAP-SAN.
- Correction de la défaillance d'importation du volume ANF avec NFS v4.1.

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 d'Amazon 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 ONTAPI (ZAPI) 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 lors des opérations de transfert 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 for NetApp 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.
- Meilleures 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 DataLIFs ne doivent plus être spécifiées lors de la configuration des `ontap-san` pilotes ou `ontap-san-economy`.

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").
- Correction du problème SLM spécifique aux systèmes back-end SAN ONTAP pour garantir que seul un sous-ensemble de LIFs 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 DataLIFs IPv6 SAN ONTAP fonctionnent désormais si elles sont spécifiées avec des crochets.
- 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/Red Hat Enterprise Linux CoreOS (RHCOS), qui nécessitent le `discard` "option de montage"; assurez-vous que la mise au rebut de mountOption est incluse dans votre^] pour prendre en charge le blocage en ligne jeter[`StorageClass`.
- 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'utilisez pas Trident 26.02, la documentation des versions précédentes est disponible sur la base du "Cycle de vie du support Trident".

- "Trident 25.10"
- "Trident 25.06"
- "Trident 25.02"
- "Trident 24.10"
- "Trident 24.06"
- "Trident 24.02"
- "Trident 23.10"
- "Trident 23.07"
- "Trident 23.04"
- "Trident 23.01"

NetApp Trident prise en charge de Trident pour les systèmes de stockage ONTAP ASA r2

NetApp Trident 25.02 et versions ultérieures prennent en charge les systèmes NetApp ASA r2 comme backends de stockage. Consultez "Systèmes ASA r2" pour plus d'informations.

ASA r2 systems require the `ontap-san` pilote. Trident ne prend pas en charge le `ontap-san-economy` pilote pour les systèmes ASA r2.

Lorsque vous spécifiez `ontap-san` comme `storageDriverName` dans la configuration du backend, Trident détecte automatiquement le système de stockage ASA r2.

Trident offre une protection des données limitée pour les systèmes ASA r2 avec Trident protect.

Les protocoles SAN pris en charge dépendent de votre version de Trident :

- Trident 25.02 et versions ultérieures prennent en charge iSCSI.
- Trident 25.06 et versions ultérieures prennent en charge NVMe/TCP en plus d'iSCSI.
- Trident 25.10 et versions ultérieures prennent en charge FC en plus d'iSCSI et NVMe/TCP.

Vous devez affecter au moins un agrégat à la machine virtuelle de stockage (SVM) pour le stockage backend ONTAP. Consultez "Affectation d'agrégats aux SVM dans les systèmes ASA r2" pour obtenir des instructions.

Opérations prises en charge

- Provisionnement de volumes persistants (PVs)
- Provisionnement dynamique des volumes
- Création et suppression de volumes
- Clonage de volumes
- Extension des volumes
- Gestion des classes de stockage

Opérations non prises en charge

- Cryptage LUKS
- SnapMirror volume réplication
- Limiter l'utilisation de l'agrégat
- Mode de réservation d'espace
- Snapshots
- Hiérarchisation

Pour plus d'informations, consultez "Options et exemples de configuration des SAN ONTAP".

Problèmes connus

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

Les problèmes connus suivants ont une incidence sur la version actuelle :

VolumeSnapshots n'atteignent pas l'état ReadyToUse

Il existe un problème connu avec le `csi-snapshotter` sidecar. Dans toutes les versions de Kubernetes, VolumeGroupSnapshots `v1beta1` empêche VolumeSnapshots d'atteindre l' `ReadyToUse` état.

Deux solutions de contournement sont disponibles :

1. Supprimez les CRD VolumeGroupSnapshots pour désactiver VolumeGroupSnapshots, puis réinstallez Trident.
2. Installez VolumeGroupSnapshots `v1beta2` et `snapshot-controller` la version 8.4.0 ou ultérieure, puis réinstallez Trident. VolumeGroupSnapshots ne fonctionnent pas sur les versions de Kubernetes antérieures à la v1.34.

La restauration des sauvegardes Restic de fichiers volumineux peut échouer

Lors de la restauration de fichiers de 30 Go ou plus à partir d'une sauvegarde Amazon S3 effectuée à l'aide de Restic, l'opération de restauration peut échouer. En guise de solution de contournement, sauvegardez les données en utilisant Kopia comme outil de transfert de données (Kopia est l'outil de transfert de données par défaut pour les sauvegardes). Se référer à "Protégez les applications à l'aide de Trident Protect" pour les

instructions.

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 la consommation et la gestion des ressources de stockage sur toutes les plateformes de stockage NetApp populaires, dans le cloud public ou sur site, y compris les clusters ONTAP sur site (AFF, FAS et ASA), ONTAP Select, Cloud Volumes ONTAP, Element software (NetApp HCI, SolidFire), Azure NetApp Files, Amazon FSx for NetApp ONTAP et Google Cloud NetApp Volumes.

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".

Plateformes Kubernetes prises en charge

Trident est compatible avec un large éventail de distributions et de plateformes Kubernetes.

Les plateformes prises en charge incluent : * Kubernetes en amont * Red Hat OpenShift * SUSE Harvester 1.7.0 (ONTAP iSCSI)

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 disposent de configurations 100 % Flash, hybrides ou 100 % HDD et proposent de nombreux modèles de déploiement : FAS sur site, baies 100 % Flash et clusters ASA, ONTAP Select et 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.

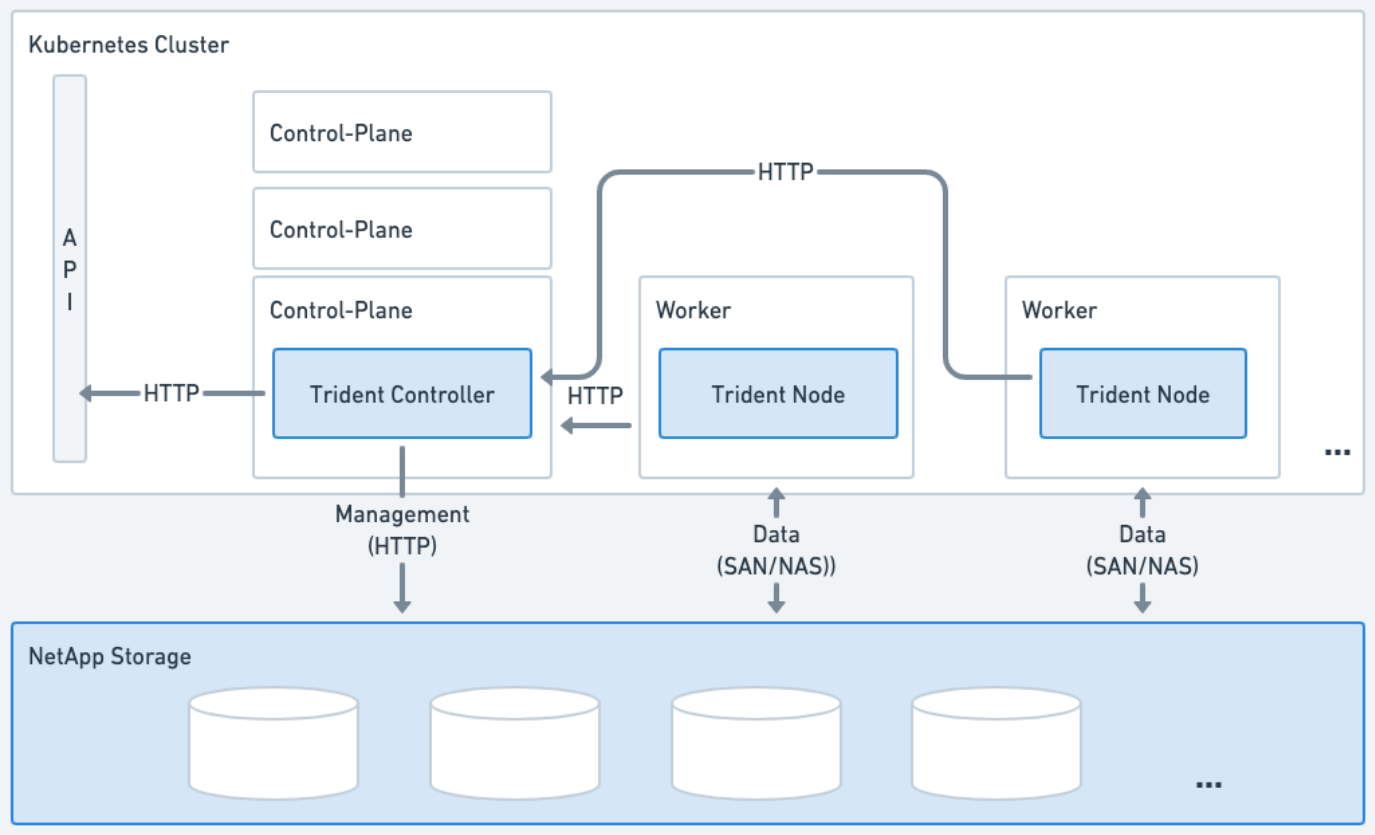


Figure 1. Trident a été déployé sur le cluster Kubernetes

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.

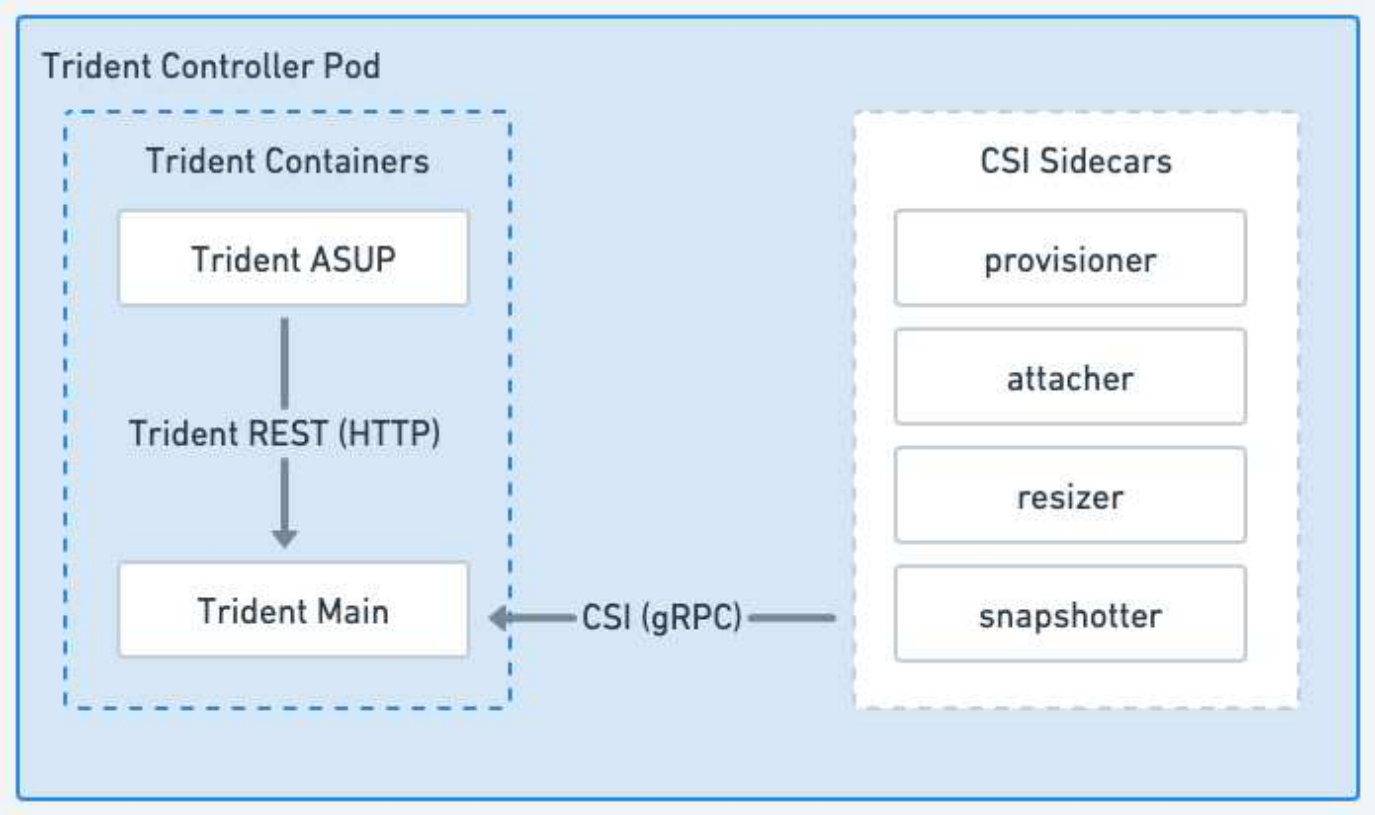


Figure 2. Diagramme du module de contrôleur Trident

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

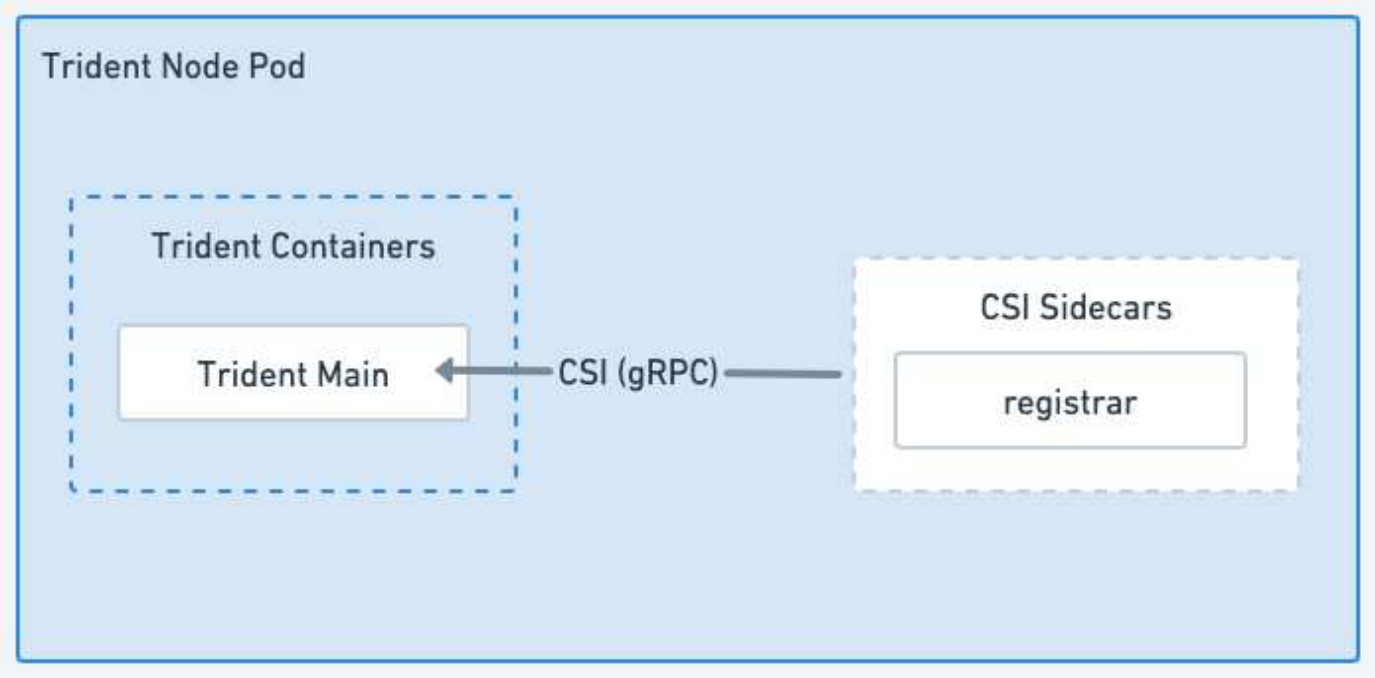


Figure 3. Diagramme Trident Node Pod

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` , et `azure-netapp-files` pilotes, chaque volume persistant (PV) est mappé à un FlexVol volume. Par conséquent, les instantanés de volume sont créés en tant qu'instantanés NetApp . La technologie de snapshots de NetApp offre une stabilité, une évolutivité, une capacité de récupération et des performances supérieures aux technologies de snapshots concurrentes. Ces copies instantanées sont extrêmement efficaces, tant 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` pilote, les volumes persistants sont mappés aux LUN créées sur des volumes FlexVol partagés. Les volumes Volumesnapshots des volumes persistants sont atteints en exécutant des copies FlexClone 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.

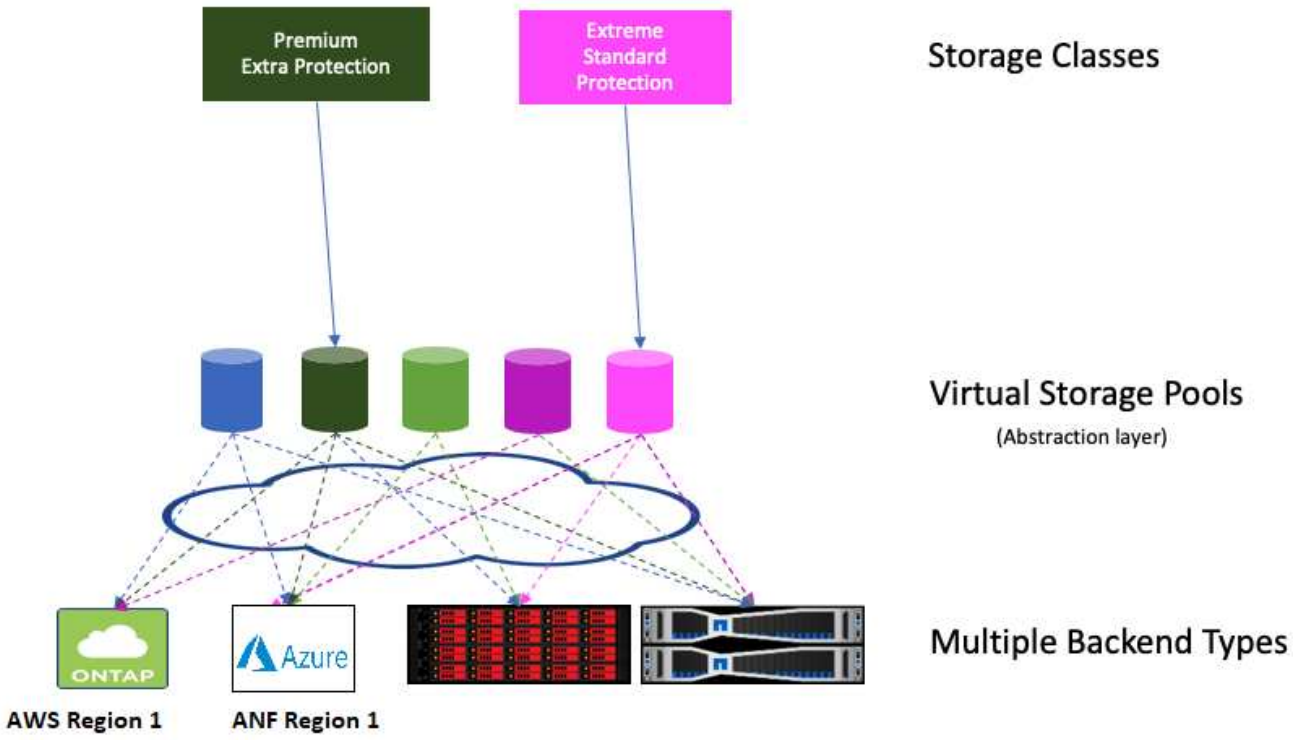
Avec Trident, vous pouvez utiliser VolumeSnapshots pour créer de nouveaux PV à partir de ceux-ci. La création de PV à partir de ces instantanés est effectuée à l'aide de la technologie FlexClone pour les backends ONTAP pris en charge. Lors de la création d'un PV à partir d'un instantané, le volume de support est un FlexClone du volume parent de l'instantané. Le `solidfire-san` Le pilote utilise les clones de volumes logiciels Element pour créer des PV à partir d'instantanés. Ici, il crée un clone à partir de l'instantané de l'élément.

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.

Les étiquettes des pools virtuels peuvent être définies à l'aide de ces caractères :

- lettres majuscules `A-Z`
- lettres minuscules `a-z`
- Nombres `0-9`
- souligne _
- traits d'union -

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 Préparez le nœud worker

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

"Préparez le nœud de travail"

2 Installez Trident

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

"Installation de Trident"

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 1.36 est désormais pris en charge dans Trident. Mettez à niveau Trident avant de mettre à niveau 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 1.27 - 1.36
- OpenShift 4.12, 4.14 - 4.22 (Si vous prévoyez d'utiliser la préparation de nœud iSCSI avec OpenShift 4.19, la version minimale de Trident prise en charge est 25.06.1.)

> Trident continue de prendre en charge les anciennes versions d'OpenShift conformément à la"Cycle de vie des versions de Red Hat Extended Update Support (EUS)" , même s'ils s'appuient sur des versions de Kubernetes qui ne sont plus officiellement prises en charge en amont. Lors de l'installation de Trident dans de tels cas, vous pouvez ignorer en toute sécurité tous les messages d'avertissement concernant la version de Kubernetes.

- Rancher Kubernetes Engine 2 (RKE2) v1.28.x - 1.36.x

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 1.25 vers la version 1.26 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
- Baie SAN 100 % Flash (ASA) de NetApp
- Systèmes FAS, AFF ou ASA r2 sur site (iSCSI, NVMe/TCP et FC) exécutant des versions ONTAP sous support complet ou limité de NetApp . Voir "Prise en charge des versions logicielles".
- Logiciel NetApp HCI/Element 11 ou version ultérieure

Prise en charge de Trident pour KubeVirt et OpenShift Virtualization

Pilotes de stockage pris en charge :

Trident prend en charge les pilotes ONTAP suivants pour KubeVirt et OpenShift Virtualization :

- ontap-nas
- san ONTAP (iSCSI, FCP, NVMe over TCP)
- ONTAP-san-Economy (iSCSI uniquement)

Points à prendre en compte :

- Mettez à jour la classe de stockage pour avoir le `fsType` paramètre (par exemple : `fsType: "ext4"`) dans l'environnement OpenShift Virtualization. Si nécessaire, définissez le mode de volume pour bloquer explicitement à l'aide du `volumeMode=Block` paramètre dans `dataVolumeTemplates` pour avertir CDI de créer des volumes de données de bloc.
- Mode d'accès *RWX pour les pilotes de stockage en mode bloc* : les pilotes ONTAP-san (iSCSI, NVMe/TCP, FC) et ONTAP-san-Economy (iSCSI) sont pris en charge uniquement avec les pilotes « volumeMode : bloc » (périphérique brut). Pour ces pilotes, le `fstype` paramètre ne peut pas être utilisé car les volumes sont fournis en mode périphérique brut.
- Pour les flux de travail de migration dynamique où le mode d'accès RWX est requis, ces combinaisons sont prises en charge :
  - NFS + `volumeMode=Filesystem`
  - ISCSI + `volumeMode=Block` (périphérique brut)
  - NVMe/TCP + `volumeMode=Block` (périphérique brut)
  - FC + `volumeMode=Block` (périphérique brut)

Prise en charge de Trident pour Kubernetes sur Kubernetes et les plans de contrôle hébergés OpenShift

À partir de la version 26.06, Trident prend pleinement en charge la topologie Kubernetes sur Kubernetes. Les clusters Kubernetes dont les nœuds s'exécutent en tant que machines virtuelles KubeVirt sur un cluster d'infrastructure (hôte). Cette topologie sous-tend les offres de plan de contrôle hébergé telles que les plans de contrôle hébergés (HCP) OpenShift sur OpenShift Virtualization.

Cette configuration fonctionnait dans les versions précédentes, mais la qualification complète commence avec 26.06.

Le cluster invité (locataire) utilise le pilote CSI KubeVirt (`kubevirt-csi` comme fournisseur de stockage. Lorsqu'une charge de travail dans le cluster invité requiert un `PersistentVolumeClaim` volume, le pilote CSI KubeVirt crée un volume correspondant dans le cluster d'infrastructure. Trident provisionne le volume à partir d'ONTAP, et le volume est connecté à chaud à la machine virtuelle du nœud invité. Trident s'exécute uniquement dans le cluster d'infrastructure. Le cluster invité consomme le stockage ONTAP indirectement via le pilote CSI KubeVirt.

Exigences :

- Trident est installé sur le cluster d'infrastructure avec un backend ONTAP pris en charge.
- Une classe de stockage Trident existe dans le cluster d'infrastructure pour que le pilote CSI KubeVirt puisse mapper les volumes du cluster invité.
- KubeVirt ou OpenShift Virtualization est déployé sur le cluster d'infrastructure, et le pilote CSI KubeVirt est installé dans chaque cluster invité.

Composant	Version
Trident	26,06
Kubernetes (infrastructure et cluster invité)	1,36
KubeVirt	1.9.0-beta.0
Importateur de données conteneurisé (CDI)	1,65

Points à prendre en compte :

- Dans le cluster invité, les volumes apparaissent comme des périphériques de stockage par blocs connectés à chaud sur les machines virtuelles des nœuds invités.
- Le flux de travail du pilote CSI KubeVirt ne nécessite pas Trident dans le cluster invité. Vous configurez le backend ONTAP et la classe de stockage dans le cluster d'infrastructure.

Installez Trident directement dans le cluster invité :

Vous pouvez également installer Trident directement dans le cluster invité (tenant). Installez Trident dans le cluster invité lorsque vous avez besoin de l'ensemble des fonctionnalités de Trident, telles que les instantanés, les clones, le redimensionnement de volumes et l'importation, que `kubevirt-csi` ne fournit pas à lui seul.

Dans le cluster invité, utilisez les protocoles IP suivants : NFS, iSCSI et NVMe/TCP. Le cluster invité ne prend pas en charge Fibre Channel (FC) car les nœuds invités sont des machines virtuelles sans connectivité FC directe.

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	1.27 - 1.36	Non
Snapshots de volume	1.27 - 1.36	Non
Volume persistant à partir des copies Snapshot des volumes	1.27 - 1.36	Non
Redimensionnement PV iSCSI	1.27 - 1.36	Non
Chap bidirectionnel ONTAP	1.27 - 1.36	Non
Règles d'exportation dynamiques	1.27 - 1.36	Non
Opérateur Trident	1.27 - 1.36	Non
Topologie CSI	1.27 - 1.36	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 Enterprise Linux CoreOS (RHCOS) prises en charge par OpenShift Container Platform sur AMD64 et ARM64
- Red Hat Enterprise Linux (RHEL) 8 ou version ultérieure sur AMD64 et ARM64

NVMe/TCP requiert RHEL 9 ou version ultérieure.

- Ubuntu 22.04 LTS ou version ultérieure sur AMD64 et ARM64
- Windows Server 2022
- SUSE Linux Enterprise Server (SLES) 15 ou version ultérieure

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.

Images de conteneur requises pour Trident 26.02

Versions de Kubernetes	Image de conteneur
v1.27.0, v1.28.0, v1.29.0, v1.30.0, v1.31.0, v1.32.0, v1.33.0, v1.34.0, v1.35.0, v1.36.0	• docker.io/netapp/trident:26.02.0 • docker.io/netapp/trident-autosupport:26.02 • registry.k8s.io/sig-storage/csi-provisioner:v6.1.0 • registry.k8s.io/sig-storage/csi-attacher:v4.10.0 • registry.k8s.io/sig-storage/csi-resizer:v2.0.0 • registry.k8s.io/sig-storage/csi-snapshotter:v8.5.0 • registry.k8s.io/sig-storage/csi-node-driver-registrar:v2.15.0 • docker.io/netapp/trident-operator:26.02.0 (facultatif)

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 essentielles concernant Trident 26.02

Vous devez lire les informations critiques suivantes sur Trident.

** informations sur le Trident **

- Kubernetes est désormais pris en charge dans Trident. Mettez à niveau Trident avant de mettre à niveau 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 :

- Accès complet à un cluster Kubernetes compatible, exécutant une version compatible de Kubernetes et dont les fonctionnalités requises sont activées. Consultez la "de formation" pour plus de détails.
- 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".
- Le cluster doit prendre en charge les workloads privilégiés.

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

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

Utilisation de l'opérateur Trident

Que ce soit par déploiement manuel ou via 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" en utilisant 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 un opérateur et que vous souhaitez utiliser le déploiement de l'opérateur Trident basé sur Helm, vous devez d'abord désinstaller manuellement l'opérateur, puis effectuer l'installation de Helm. Cela permet à Helm de déployer l'opérateur Trident avec les étiquettes et annotations requises. Si vous ne le faites pas, votre déploiement d'opérateur Trident basé sur Helm échouera avec une erreur de validation d'étiquette et une erreur de validation d'annotation.
- Si vous avez un `tridentctl` Déploiement basé sur Helm, vous pouvez effectuer un déploiement basé sur Helm ou sur Operator sans désinstaller Trident.

Autres options de configuration connues

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

- 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 essentielles concernant Trident 26.02

Vous devez lire les informations critiques suivantes sur Trident.

** informations sur le Trident **

- Kubernetes 1.35 est désormais pris en charge dans Trident. Mettez à niveau Trident avant de mettre à niveau 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

Vérifiez "présentation de l'installation" pour vous assurer que vous avez respecté les prérequis d'installation et sélectionné la bonne option d'installation pour votre environnement.

Avant de commencer

Avant de commencer l'installation, connectez-vous à l'hôte Linux et vérifiez qu'il gère un système fonctionnel, "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/v26.02.0/trident-
installer-26.02.0.tar.gz
tar -xf trident-installer-26.02.0.tar.gz
cd trident-installer
```

Étape 2 : créez le `TridentOrchestrator` CRD

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

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

Étape 3 : déployer l'opérateur Trident

Le programme d'installation 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:26.02
    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:26.02.0
  Message:                  Trident installed  Namespace:
trident
  Status:                   Installed
  Version:                  v26.02.0
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 |
+----------------+----------------+
| 26.02.0        | 26.02.0        |
+----------------+----------------+
```

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

Vous devez lire les informations critiques suivantes sur Trident.

** informations sur le Trident **

- Kubernetes 1.36 est désormais pris en charge dans Trident. Mettez à niveau Trident avant de mettre à niveau 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

Vérifiez "présentation de l'installation" pour vous assurer que vous avez respecté les prérequis d'installation et sélectionné la bonne option d'installation pour votre environnement.

Avant de commencer

Connectez-vous à l'hôte Linux et vérifiez qu'il gère un système fonctionnel et "Cluster Kubernetes pris en charge" 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/v6.0/trident-
installer-26.02.0.tar.gz
tar -xf trident-installer-26.02.0.tar.gz
cd trident-installer
```

Étape 2 : créez le `TridentOrchestrator` CRD

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

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

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

Dans `/deploy/operator.yaml`, mettez à jour `image: docker.io/netapp/trident-operator:26.02.0` 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 être situées dans le même registre. Par exemple :

- `image: <your-registry>/trident-operator:26.02.0` si toutes vos images sont situées dans un seul registre.

- `image: <your-registry>/netapp/trident-operator:26.02.0` si votre image Trident se trouve dans un registre différent de celui 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:26.02"
tridentImage: "<your-registry>/trident:26.02.0"
```

Images dans différents registres

```
imageRegistry: "<your-registry>"
autosupportImage: "<your-registry>/trident-autosupport:26.02"
tridentImage: "<your-registry>/trident:26.02.0"
```

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:26.02
  Debug:              true
  Image Registry:     <your-registry>
  Namespace:          trident
  Trident Image:      <your-registry>/trident:26.02.0
Status:
  Current Installation Params:
    IPv6:                       false
    Autosupport Hostname:
    Autosupport Image:          <your-registry>/trident-autosupport:26.02
    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:26.02.0
  Message:                Trident installed
  Namespace:              trident
  Status:                 Installed
  Version:                v26.02.0
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 |
+----------------+----------------+
| 26.02.0        | 26.02.0        |
+----------------+----------------+
```

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 essentielles concernant le Trident 25.10

Vous devez lire les informations critiques suivantes sur Trident.

informations sur le Trident

- Kubernetes 1.35 est désormais pris en charge dans Trident. Mettez à niveau Trident avant de mettre à niveau 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.

Vérifiez "présentation de l'installation" pour vous assurer que vous avez respecté les prérequis d'installation et sélectionné la bonne option d'installation pour votre environnement.

Avant de commencer

En plus de "conditions préalables au déploiement" vous avez besoin de "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..0` est la version de Trident que vous installez.

```
helm install <name> netapp-trident/trident-operator --version 100.2602.0
--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 `100.2602.0` est la version de Trident que vous installez :

```
helm install <name> netapp-trident/trident-operator --version 100.2602.0
--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 journalisation de Trident à `debug` . Vous pouvez automatiser le processus de détachement forcé grâce à l'intégration avec l'opérateur de vérification de l'état du nœud (NHC). Pour plus d'informations, voir"Automatisation du basculement des applications avec état grâce à Trident" .	`false`
`tridentLogWorkflows`	Permet d'activer des flux de travail Trident spécifiques pour la consignation des traces ou la suppression des journaux.	`""`

Option	Description	Valeur par défaut
`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.	`""`
`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`

Option	Description	Valeur par défaut
`nodePrep`	Permet à Trident de préparer les nœuds du cluster Kubernetes pour gérer les volumes à l'aide du protocole de stockage de données spécifié. **Actuellement, `iscsi` est la seule valeur prise en charge.** REMARQUE : à partir d'OpenShift 4.19, la version minimale de Trident prise en charge pour cette fonctionnalité est 25.06.1.	
`enableConcurrency`	Permet des opérations simultanées du contrôleur Trident pour un débit amélioré. **Aperçu technique** : Cette fonctionnalité est expérimentale et prend actuellement en charge des flux de travail parallèles limités avec les pilotes ONTAP-NAS (NFS uniquement) et ONTAP-SAN (NVMe pour ONTAP 9 unifié), en plus de l'aperçu technique existant pour le pilote ONTAP-SAN (protocoles iSCSI et FCP dans ONTAP 9 unifié).	faux
`k8sAPIQPS`	La limite de requêtes par seconde (QPS) utilisée par le contrôleur lors de la communication avec le serveur API Kubernetes. La valeur Burst est définie automatiquement en fonction de la valeur QPS.	`100`; facultatif

Option	Description	Valeur par défaut

`resources`	Définit les limites et les requêtes de ressources Kubernetes pour le contrôleur Trident , le nœud et les pods d'opérateur. Vous pouvez configurer le processeur et la mémoire de chaque conteneur et sidecar pour gérer l'allocation des ressources dans Kubernetes.
	
	Pour plus d'informations sur la configuration des demandes et des limites de ressources, veuillez consulter la documentation."Gestion des ressources pour les pods et les conteneurs" .
	
	• NE MODIFIEZ PAS les noms des conteneurs ou des champs.
	• NE PAS modifier l'indentation - L'indentation YAML est essentielle pour une analyse correcte.
		<pre>resources:
		  controller:
		    trident-main:
		      requests:
		        cpu: 10m
		        memory: 80Mi
		      limits:
		          cpu:
		          memory:
		    csi-provisioner:
		      requests:
		        cpu: 2m
		        memory: 20Mi
		      limits:
		        cpu:
		        memory:
		    csi-attacher:
		      requests:
		        cpu: 2m
		        memory: 20Mi
		      limits:
		        cpu:
		        memory:
		    csi-resizer:
		      requests:
		        cpu: 3m
		        memory: 20Mi
		      limits:
		        cpu:
		        memory:
		    csi-snapshotter:
		      requests:
		        cpu: 2m
		        memory: 20Mi
		      limits:
		        cpu:
		        memory:
		    trident-autosupport:
		      requests:
		        cpu: 1m
		        memory: 30Mi
		      limits:
		        cpu:
		        memory:
		  node:
		    linux:
		      trident-main:</pre>

Option	Description	Valeur par défaut
`httpsMetrics`	Activer HTTPS pour le point de terminaison des métriques Prometheus.	faux
`hostNetwork`	Permet la mise en réseau de l'hôte pour le contrôleur Trident . Ceci est utile lorsque vous souhaitez séparer le trafic frontal et le trafic dorsal dans un réseau multi-foyers.	faux

t si elles ne sont pas spécifiées.

- Les noms des conteneurs sont listés tels qu'ils apparaissent dans les spécifications du pod.
- Les side-cars sont répertoriés sous chaque conteneur principal.
- Vérifiez les TORC `status.CurrentInstallationParams` permettant d'afficher les valeurs actuellement appliquées.

```
            cpu: 1m
            memory: 10Mi
          limits:
            cpu:
            memory:
      windows:
        trident-main:
          requests:
            cpu: 6m
            memory: 40Mi
          limits:
            cpu:
            memory:
        node-driver-registrar:
          requests:
            cpu: 6m
            memory: 40Mi
          limits:
            cpu:
            memory:
        liveness-probe:
          requests:
            cpu: 2m
            memory: 40Mi
          limits:
            cpu:
            memory:
    operator:
      requests:
        cpu: 10m
        memory: 40Mi
      limits:
        cpu:
        memory:
```

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 essentielles concernant Trident 26.02

Vous devez lire les informations critiques suivantes sur Trident.

** informations sur le Trident **

- Kubernetes 1.35 est désormais pris en charge dans Trident. Mettez à niveau Trident avant de mettre à niveau 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.

Vérifiez "présentation de l'installation" pour vous assurer que vous avez respecté les prérequis d'installation et sélectionné la bonne option d'installation pour votre environnement.

Avant de commencer

En plus de "conditions préalables au déploiement" vous avez besoin de "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 être situées dans le même registre. Dans les exemples, `100.2602.0` correspond à la version de Trident que vous installez.

Images dans un registre

```
helm install <name> netapp-trident/trident-operator --version
100.2602.0 --set imageRegistry=<your-registry> --create-namespace
--namespace <trident-namespace> --set nodePrep={iscsi}
```

Images dans différents registres

```
helm install <name> netapp-trident/trident-operator --version
100.2602.0 --set imageRegistry=<your-registry> --set operatorImage
=<your-registry>/trident-operator:26.02.0 --set
tridentAutosupportImage=<your-registry>/trident-autosupport:26.02
--set tridentImage=<your-registry>/trident:26.02.0 --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 `100.2602.0` est la version de Trident que vous installez :

```
helm install <name> netapp-trident/trident-operator --version 100.2602.0
--create-namespace --namespace trident --set tridentDebug=true
```

Pour ajouter la valeur nodePrep, exécutez la commande suivante :

```
helm install <name> netapp-trident/trident-operator --version 100.2602.0
--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 180 secondes pour la plupart des opérations d'API Kubernetes (si différent de zéro, en secondes). Le `tridentK8sTimeout` le paramètre s'applique uniquement à l'installation de Trident .	`180`
`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`

Option	Description	Valeur par défaut
`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.	« »
`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 de détachement forcé. Vous pouvez automatiser le processus de détachement forcé grâce à l'intégration avec l'opérateur de vérification de l'état du nœud (NHC). Pour plus d'informations, voir"Automatisation du basculement des applications avec état grâce à Trident" .	`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.** À partir d'OpenShift 4.19, la version minimale de Trident prise en charge pour cette fonctionnalité est 25.06.1.	

Option	Description	Valeur par défaut

`resources`	Définit les limites et les requêtes de ressources Kubernetes pour le contrôleur Trident , le nœud et les pods d'opérateur. Vous pouvez configurer le processeur et la mémoire de chaque conteneur et sidecar pour gérer l'allocation des ressources dans Kubernetes.
	
	Pour plus d'informations sur la configuration des demandes et des limites de ressources, veuillez consulter la documentation."Gestion des ressources pour les pods et les conteneurs" .
	
	• NE MODIFIEZ PAS les noms des conteneurs ou des champs.
	• NE PAS modifier l'indentation - L'indentation YAML est essentielle pour une analyse correcte.
		`resources:`
		`  controller:`
		`  trident-main:`
		`  requests:`
		`  cpu: 10m`
		`  memory: 80Mi`
		`  limits:`
		`  cpu:`
		`  memory:`
		`  csi-provisioner:`
		`  requests:`
		`  cpu: 2m`
		`  memory: 20Mi`
		`  limits:`
		`  cpu:`
		`  memory:`
		`  csi-attacher:`
		`  requests:`
		`  cpu: 2m`
		`  memory: 20Mi`
		`  limits:`
		`  cpu:`
		`  memory:`
		`  csi-resizer:`
		`  requests:`
		`  cpu: 3m`
		`  memory: 20Mi`
		`  limits:`
		`  cpu:`
		`  memory:`
		`  csi-snapshotter:`
		`  requests:`
		`  cpu: 2m`
		`  memory: 20Mi`
		`  limits:`
		`  cpu:`
		`  memory:`
		`  trident-autosupport:`
		`  requests:`
		`  cpu: 1m`
		`  memory: 30Mi`
		`  limits:`
		`  cpu:`
		`  memory:`
		`  node:`
		`  linux:`

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 sous la forme d'un pod de contrôleur unique et d'un pod de nœud sur chaque nœud de travail du cluster. Le pod du nœud doit être en cours d'exécution sur tout 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` , `ontap-nas` , et `ontap-nas-economy` seulement. Fonctionne avec Kubernetes Non-Graceful Node Shutdown (NGNS) pour permettre aux administrateurs de cluster de migrer en toute sécurité les charges de travail avec des volumes montés vers de nouveaux nœuds si un nœud devient défaillant. Pour plus d'informations, voir"Automatisation du basculement des applications avec état grâce à Trident" .	`false`
`windows`	Réglage sur `true` Active l'installation sur les nœuds de travail Windows.	`false`

- Aucune limite n'est appliquée par défaut ; seules les requêtes ont des valeurs par défaut.
- Les noms des conteneurs sont listés tels qu'ils apparaissent dans les spécifications du pod.
- Les side-cars sont répertoriés avec chaque conteneur principal.
- Vérifiez les TORC `status.CurrentInstallationParams` champ permettant d'afficher les valeurs attributs actuellement appliquées.

```
    trident-main:
      requests:
        cpu: 10m
        memory: 60Mi
      limits:
        cpu:
        memory:
    autosupport:
    registrar:
      requests:
        cpu: 1m
        memory: 10Mi
      limits:
        cpu:
        memory:
  windows:
    trident-main:
      requests:
        cpu: 6m
        memory: 40Mi
      limits:
        cpu:
        memory:
    node-driver-
        memory:
  operator:
    requests:
      cpu: 10m
      memory: 40Mi
    limits:
```

Paramètre	Description	Valeur par défaut
`cloudProvider`	Défini sur `"Azure"` lors de l'utilisation d'identités gérées ou d'une identité cloud sur un cluster AKS. Défini sur `"AWS"` lors de l'utilisation d'une identité de cloud sur un cluster EKS. Défini sur `"GCP"` lors de l'utilisation d'une identité de nuage sur un cluster GKE.	`""`
`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. Défini sur l'identité cloud (« iam.gke.io/gcp-service-account: xxxx@mygcpproject.iam.gserviceaccount.com ») lors de l'utilisation de l'identité cloud sur un cluster GKE.	`""`
`IPv6`	Installez Trident sur IPv6	faux
`k8sTimeout`	Délai d'expiration pour les opérations Kubernetes. Le `k8sTimeout` le paramètre s'applique uniquement à l'installation de Trident .	`180sec`
`silenceAutosupport`	N'envoyez pas de packs AutoSupport à NetApp automatiquement	`false`
`autosupportImage`	Image conteneur pour la télémétrie AutoSupport	`"netapp/trident-autosupport10"`
`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:26.02"`
`imageRegistry`	Chemin d'accès au registre interne, du format `<registry FQDN>[:port][/subpath]`	`"registry.k8s.io"`
`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	

Paramètre	Description	Valeur par défaut
`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.** À partir d'OpenShift 4.19, la version minimale de Trident prise en charge pour cette fonctionnalité est 25.06.1.	
`k8sAPIQPS`	La limite de requêtes par seconde (QPS) utilisée par le contrôleur lors de la communication avec le serveur API Kubernetes. La valeur Burst est définie automatiquement en fonction de la valeur QPS.	`100`; facultatif
`enableConcurrency`	Permet des opérations simultanées du contrôleur Trident pour un débit amélioré. **Aperçu technique** : Cette fonctionnalité est expérimentale et prend actuellement en charge des flux de travail parallèles limités avec les pilotes ONTAP-NAS (NFS uniquement) et ONTAP-SAN (NVMe pour ONTAP 9 unifié), en plus de l'aperçu technique existant pour le pilote ONTAP-SAN (protocoles iSCSI et FCP dans ONTAP 9 unifié).	faux

Paramètre	Description	Valeur par défaut
`resources`	Définit les limites et les requêtes de ressources Kubernetes pour le contrôleur Trident et les pods de nœuds. Vous pouvez configurer le processeur et la mémoire de chaque conteneur et sidecar pour gérer l'allocation des ressources dans Kubernetes. Pour plus d'informations sur la configuration des demandes et des limites de ressources, veuillez consulter la documentation."Gestion des ressources pour les pods et les conteneurs" . • NE MODIFIEZ PAS les noms des conteneurs ou des champs. • NE PAS modifier l'indentation - L'indentation YAML est essentielle pour une analyse correcte. • Aucune limite n'est appliquée par défaut ; seules les requêtes ont des valeurs par défaut et sont appliquées automatiquement si elles ne sont pas spécifiées. • Les noms des conteneurs sont listés tels qu'ils apparaissent dans les spécifications du pod. • Les side-cars sont répertoriés sous chaque conteneur principal. • Vérifiez les TORC `status.CurrentInstallationParams` champ permettant d'afficher les valeurs actuellement appliquées.	`resources: controller: trident-main: requests: cpu: 10m memory: 80Mi limits: cpu: memory: csi-provisioner: requests: cpu: 2m memory: 20Mi limits: cpu: memory: csi-attacher: requests: cpu: 2m memory: 20Mi limits: cpu: memory: csi-resizer: requests: cpu: 3m memory: 20Mi limits: cpu: memory: csi-snapshotter: requests: cpu: 2m memory: 20Mi limits:`

Paramètre	Description	Valeur par défaut
`httpsMetrics`	Activer HTTPS pour le point de terminaison des métriques Prometheus.	faux
`hostNetwork`	Permet la mise en réseau de l'hôte pour le contrôleur Trident . Ceci est utile lorsque vous souhaitez séparer le trafic frontal et le trafic dorsal dans un réseau multi-foyers.	faux

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

Exemples de configurations

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

configuration personnalisée de base

Cet exemple, créé après l'exécution de `cat deploy/crds/tridentorchestrator_cr_imagepullsecrets.yaml` commande, représente une installation personnalisée de base :

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

```
30Mi
        limits:
          cpu:
          memory:
    node:
      linux:
        trident-
main:
1m
memory: 10Mi
        limits:
          cpu:
memory:
    windows:
      trident-
main:
requests:
          cpu:
6m
memory: 40Mi
        limits:
```

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
```

memory:
liveness-

nœuds de travail Windows

Cet exemple, créé après l'exécution de `cat deploy/crds/tridentorchestrator_cr.yaml` commande, installe Trident sur un nœud de travail Windows.

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

Gestion des identités 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 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 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 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
```

Configuration des requêtes et limites de ressources Kubernetes pour le contrôleur Trident et les pods de nœuds Trident Linux

Cet exemple configure les demandes et les limites de ressources Kubernetes pour le contrôleur Trident et les pods de nœuds Trident Linux.

Avertissement : Les valeurs de requête et de limite fournies dans cet exemple sont uniquement à des fins de démonstration. Ajustez ces valeurs en fonction de votre environnement et de vos exigences en matière de charge de travail.

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  imagePullSecrets:
  - thisisasecret
  resources:
    controller:
      trident-main:
        requests:
          cpu: 10m
          memory: 80Mi
        limits:
          cpu: 200m
          memory: 256Mi
      # sidecars
      csi-provisioner:
        requests:
          cpu: 2m
          memory: 20Mi
        limits:
          cpu: 100m
          memory: 64Mi
      csi-attacher:
        requests:
          cpu: 2m
          memory: 20Mi
        limits:
          cpu: 100m
          memory: 64Mi
      csi-resizer:
        requests:
          cpu: 3m
```

```
          memory: 20Mi
        limits:
          cpu: 100m
          memory: 64Mi
      csi-snapshotter:
        requests:
          cpu: 2m
          memory: 20Mi
        limits:
          cpu: 100m
          memory: 64Mi
      trident-autosupport:
        requests:
          cpu: 1m
          memory: 30Mi
        limits:
          cpu: 50m
          memory: 128Mi
    node:
      linux:
        trident-main:
          requests:
            cpu: 10m
            memory: 60Mi
          limits:
            cpu: 200m
            memory: 256Mi
        # sidecars
        node-driver-registrar:
          requests:
            cpu: 1m
            memory: 10Mi
          limits:
            cpu: 50m
            memory: 32Mi
```

Configuration des requêtes et des limites de ressources Kubernetes pour le contrôleur Trident et les pods de nœuds Trident Windows et Linux

Cet exemple configure les demandes et les limites de ressources Kubernetes pour le contrôleur Trident et les pods de nœuds Trident Windows et Linux.

Avertissement : Les valeurs de requête et de limite fournies dans cet exemple sont uniquement à des fins de démonstration. Ajustez ces valeurs en fonction de votre environnement et de vos exigences en matière de charge de travail.

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  imagePullSecrets:
  - thisisasecret
  windows: true
  resources:
    controller:
      trident-main:
        requests:
          cpu: 10m
          memory: 80Mi
        limits:
          cpu: 200m
          memory: 256Mi
        # sidecars
      csi-provisioner:
        requests:
          cpu: 2m
          memory: 20Mi
        limits:
          cpu: 100m
          memory: 64Mi
      csi-attacher:
        requests:
          cpu: 2m
          memory: 20Mi
        limits:
          cpu: 100m
          memory: 64Mi
      csi-resizer:
        requests:
```

```
          cpu: 3m
          memory: 20Mi
        limits:
          cpu: 100m
          memory: 64Mi
      csi-snapshotter:
        requests:
          cpu: 2m
          memory: 20Mi
        limits:
          cpu: 100m
          memory: 64Mi
      trident-autosupport:
        requests:
          cpu: 1m
          memory: 30Mi
        limits:
          cpu: 50m
          memory: 128Mi
    node:
      linux:
        trident-main:
          requests:
            cpu: 10m
            memory: 60Mi
          limits:
            cpu: 200m
            memory: 256Mi
        # sidecars
        node-driver-registrar:
          requests:
            cpu: 1m
            memory: 10Mi
          limits:
            cpu: 50m
            memory: 32Mi
      windows:
        trident-main:
          requests:
            cpu: 6m
            memory: 40Mi
          limits:
            cpu: 200m
            memory: 128Mi
        # sidecars
        node-driver-registrar:
```

```
          requests:
            cpu: 6m
            memory: 40Mi
          limits:
            cpu: 100m
            memory: 128Mi
        liveness-probe:
          requests:
            cpu: 2m
            memory: 40Mi
          limits:
            cpu: 50m
            memory: 64Mi
```

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 essentielles concernant Trident10

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`

Vérifiez "présentation de l'installation" pour vous assurer que vous avez respecté les prérequis d'installation et sélectionné la bonne option d'installation pour votre environnement.

Avant de commencer

Avant de commencer l'installation, connectez-vous à l'hôte Linux et vérifiez qu'il gère un système fonctionnel, "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/v26.02.0/trident-
installer-26.02.0.tar.gz
tar -xf trident-installer-26.02.0.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:26.02 --trident
-image <your-registry>/trident:26.02.0
```

Images dans différents registres

```
./tridentctl install -n trident --image-registry <your-registry>
--autosupport-image <your-registry>/trident-autosupport:26.02 --trident
-image <your-registry>/trident:26.02.0
```

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=26.10.0
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 l'installateur ne se termine pas correctement ou si `trident-controller-<generated id>` (`trident-csi-<generated id>` dans les versions antérieures à 23.01 n'a pas le statut **Running**, la plateforme n'a pas été installée. Utilisez `-d` pour "activer le mode débogage" et 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 |
+----------------+----------------+
| 26.02.0        | 26.02.0        |
+----------------+----------------+
```

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, voir"Automatisation du basculement des applications avec état grâce à Trident" .

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

Activer les opérations simultanées du contrôleur Trident

Pour activer les opérations simultanées du contrôleur Trident pour un débit amélioré, ajoutez le `--enable-concurrency` option lors de l'installation comme indiqué dans cet exemple.

> **Aperçu technique** : Cette fonctionnalité est expérimentale et prend actuellement en charge des flux de travail parallèles limités avec les pilotes ONTAP-NAS (NFS uniquement) et ONTAP-SAN (NVMe pour ONTAP 9 unifié), en plus de l'aperçu technique existant pour le pilote ONTAP-SAN (protocoles iSCSI et FCP dans ONTAP 9 unifié).

```
tridentctl install --enable-concurrency -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
```

Installation à l'aide d'un opérateur certifié OpenShift

Installez Trident à l'aide d'OpenShift OperatorHub

Si vous utilisez Red Hat OpenShift, vous pouvez installer NetApp Trident à l'aide de l'opérateur certifié Red Hat. Suivez cette procédure pour installer Trident à partir de Red Hat OpenShift Container Platform.

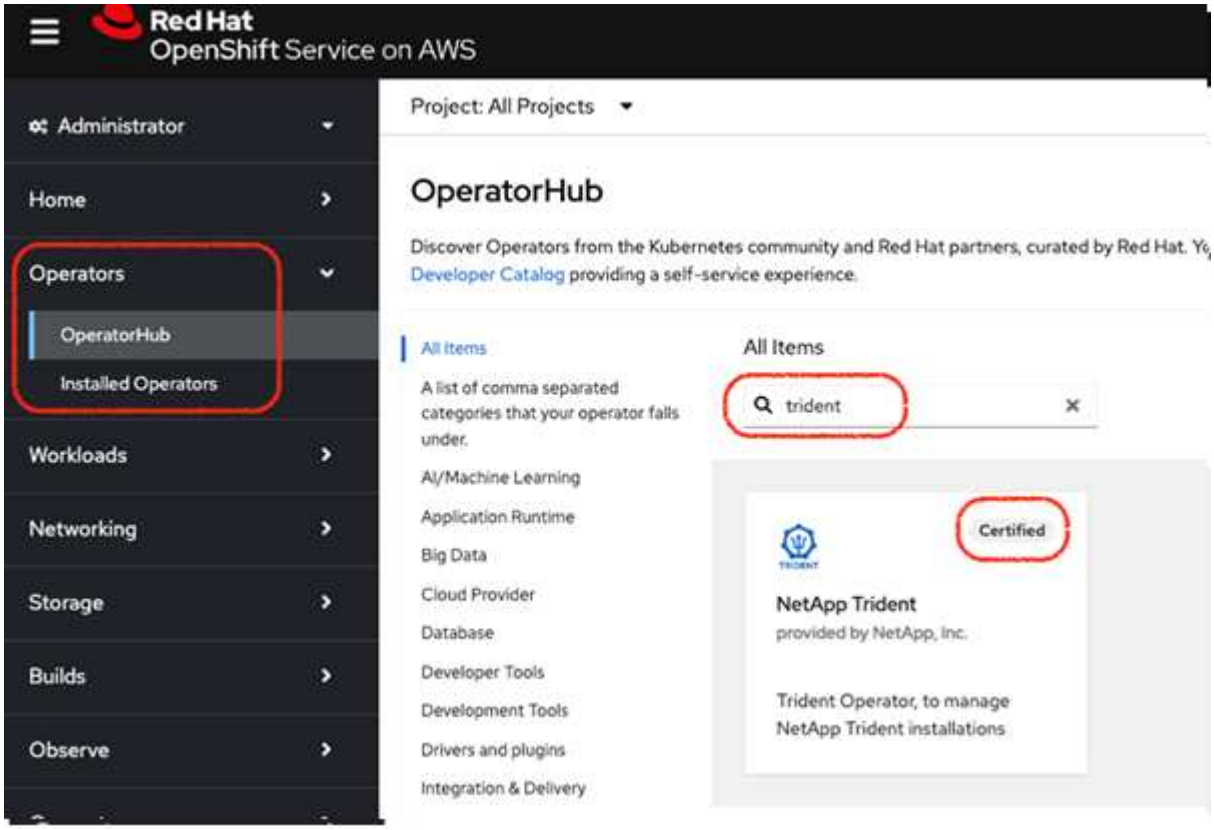
Avant de commencer

Avant de commencer l'installation, "Préparez votre environnement pour l'installation de Trident".

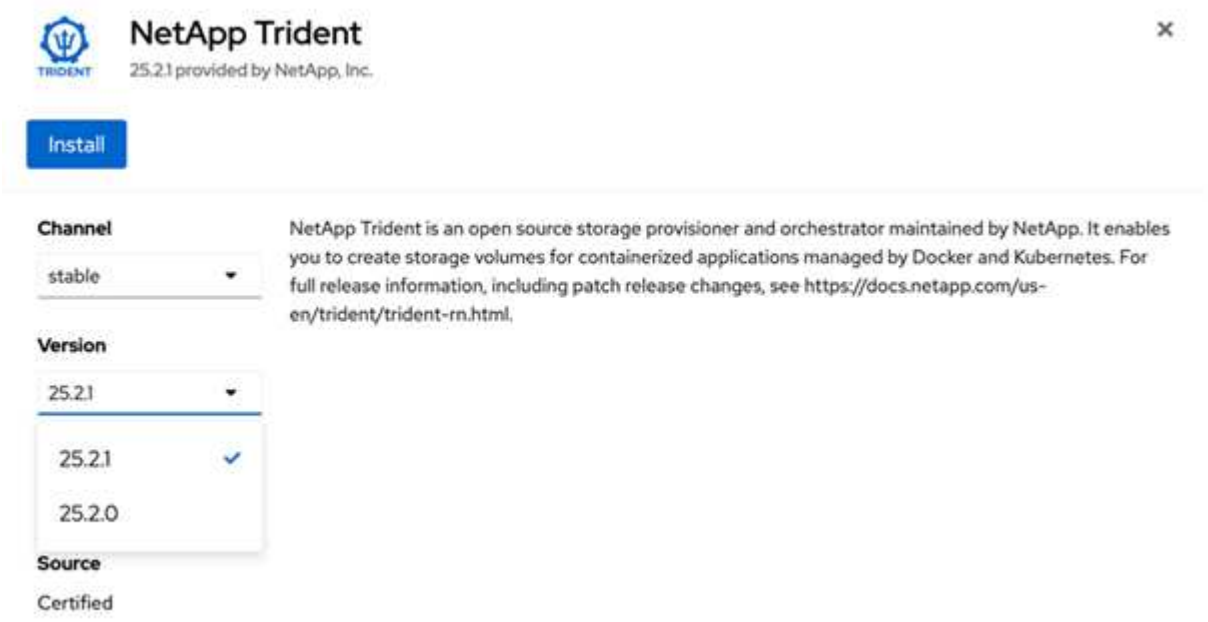
Rechercher et installer l'opérateur Trident

Étapes

1. Accédez à OpenShift OperatorHub et recherchez NetApp Trident.

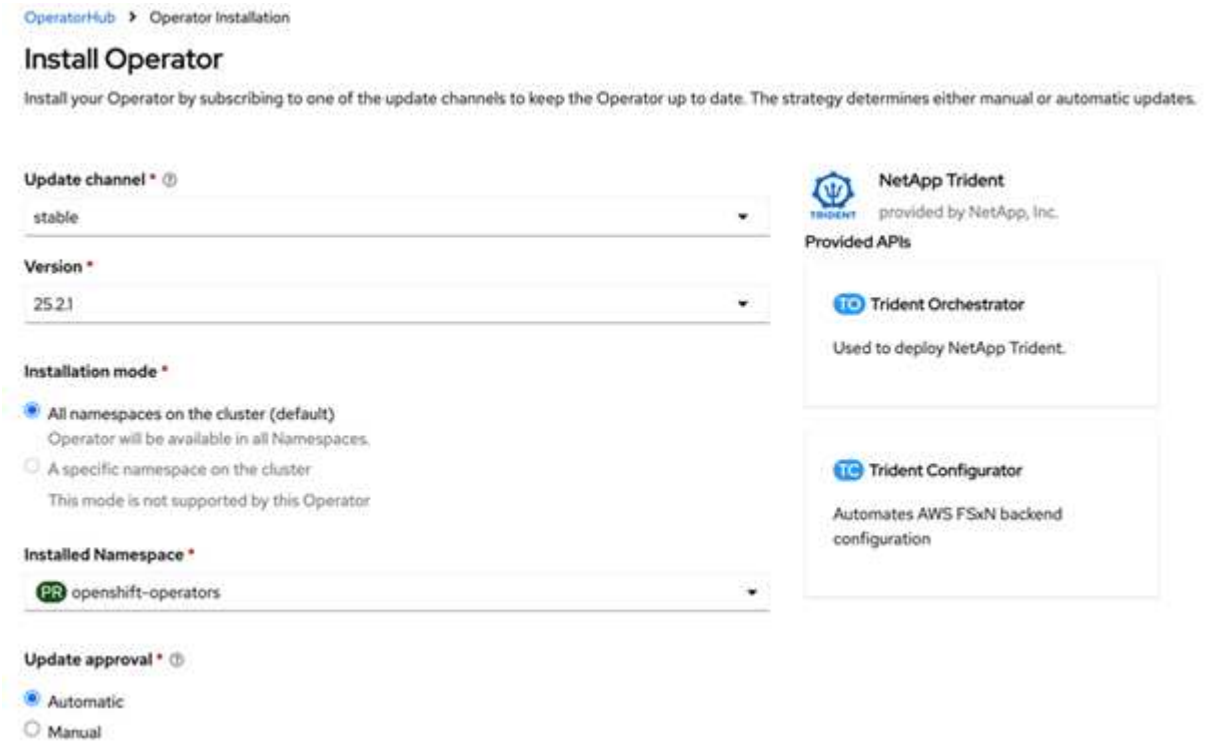


2. Cliquez sur * NetApp Trident* pour ouvrir les paramètres d'installation.
3. Sélectionnez les options requises et cliquez sur **Installer** pour ouvrir la configuration de l'opérateur.

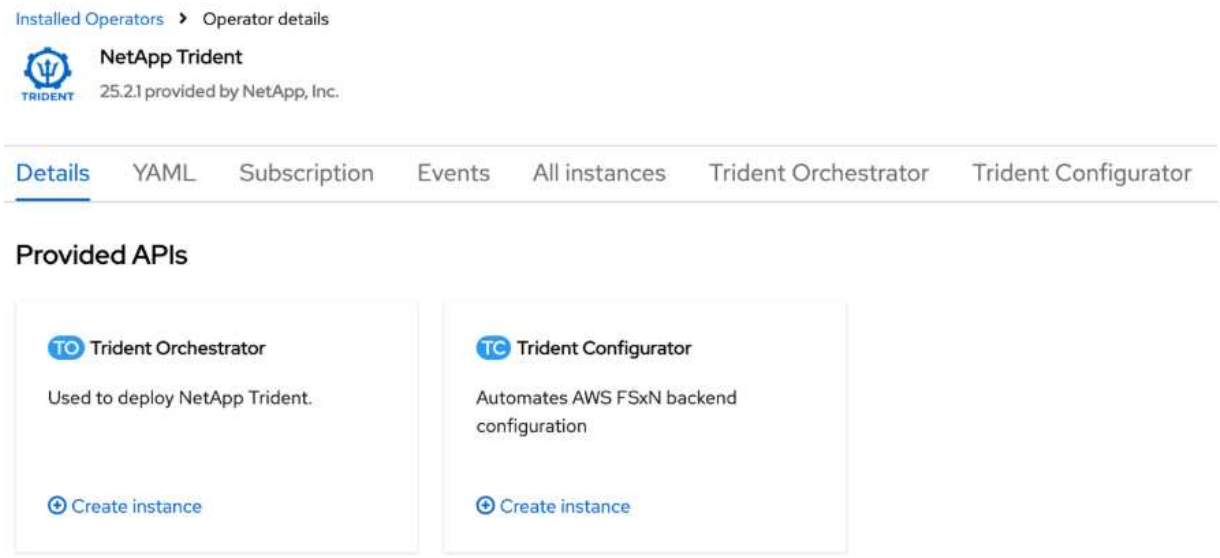


Assurez-vous de sélectionner la version la plus récente de l'opérateur.

4. Conservez tous les paramètres tels quels et cliquez sur **Installer**.



5. Cliquez sur **Afficher l'opérateur** pour afficher les détails de l'opérateur.



6. Cliquez sur **Vue YAML** et collez ce qui suit dans le formulaire :

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
  namespace: openshift-operators
spec:
  IPv6: false
  debug: false
  nodePrep:
  - iscsi
  imageRegistry: ''
  k8sTimeout: 180s
  namespace: trident
  silenceAutosupport: false
```

L'interface utilisateur propose un exemple par défaut. Vous pouvez le modifier directement au lieu de copier une configuration complète.

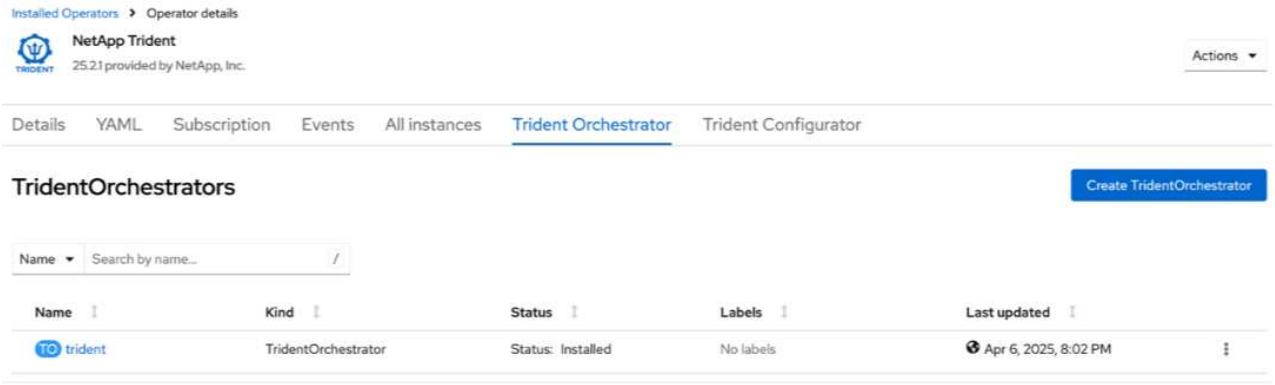
Facultatif : Activer la concurrence

Pour activer la concurrence, ajoutez le champ suivant à la spécification :

```
enableConcurrency: true
```

- Red Hat Enterprise Linux CoreOS (RHCOS) n'a pas iSCSI activé et configuré.
- Vous pouvez ajouter le `nodePrep` paramètre permettant de configurer et d'activer les services iSCSI et Multipath sur tous les nœuds de travail OpenShift.
- À partir d'OpenShift 4.19, la version minimale de Trident prise en charge pour cette fonctionnalité est 25.06.1.

1. Cliquez sur **Créer** ; Trident Orchestrator sera entièrement installé.



Désinstaller l'opérateur Trident

Étapes

1. Sélectionnez l'opérateur Trident dans la liste des opérateurs installés.
2. Sélectionnez cette option pour supprimer toutes les instances d'opérande de l'opérateur.

Si vous ne cochez pas la case **Supprimer toutes les instances d'opérande de cet opérateur**, Trident ne sera pas désinstallé.

3. Cliquez sur **Désinstaller**.

Passez à l'opérateur Trident certifié OpenShift

Vous pouvez passer à l'opérateur Trident certifié Red Hat OpenShift depuis l'opérateur communautaire, une installation basée sur Helm ou un opérateur déployé manuellement. Le processus pour chaque méthode consiste à désinstaller l'opérateur existant, puis à installer l'opérateur certifié en utilisant le OperatorHub.

Avant de commencer

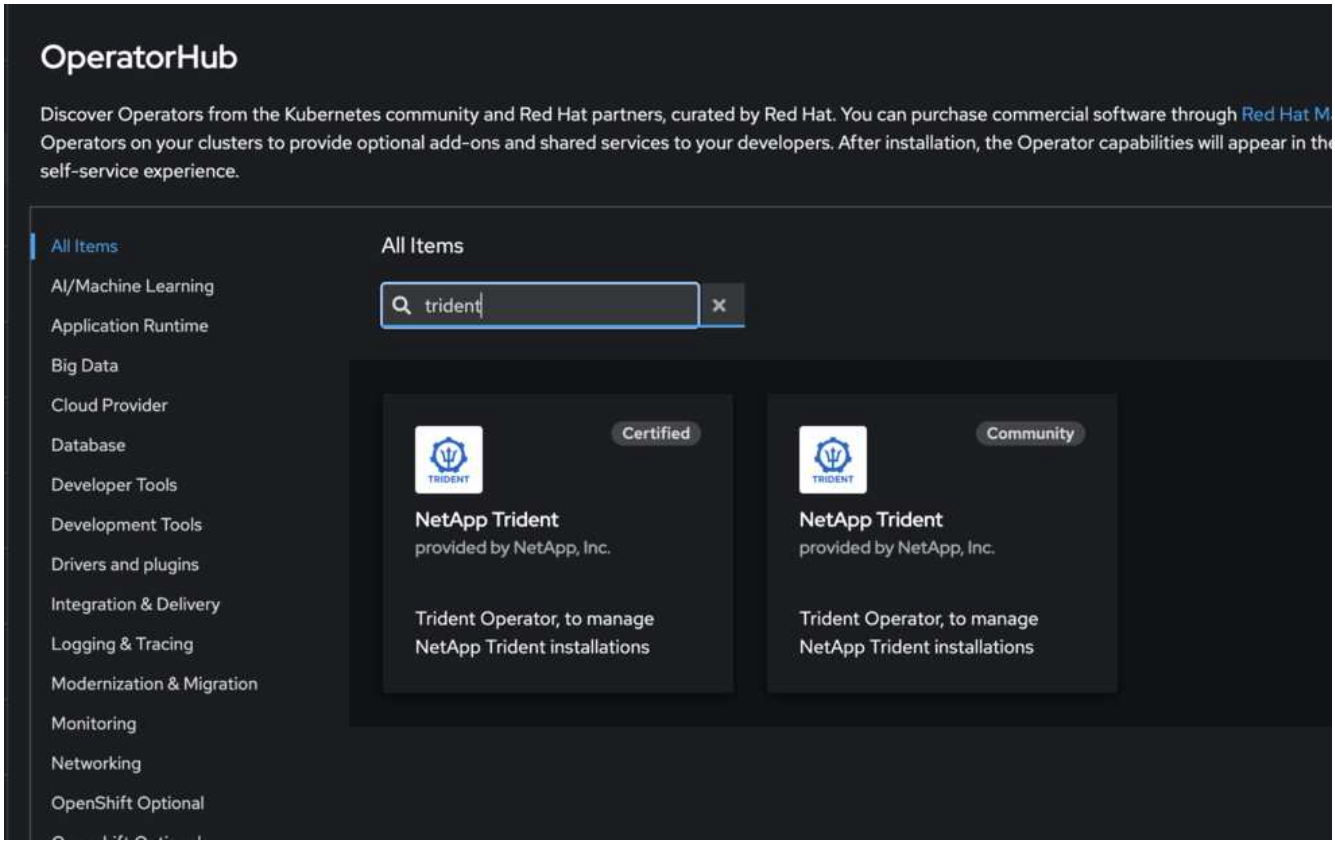
Avant de commencer l'installation, "Préparez votre environnement pour l'installation de Trident".

Ne supprimez pas la `TridentOrchestrator` ressource personnalisée (CR) lors du processus de désinstallation. La `TridentOrchestrator` CR conserve la configuration de votre backend et de votre classe de stockage, ce qui est requis après l'installation de l'opérateur certifié.

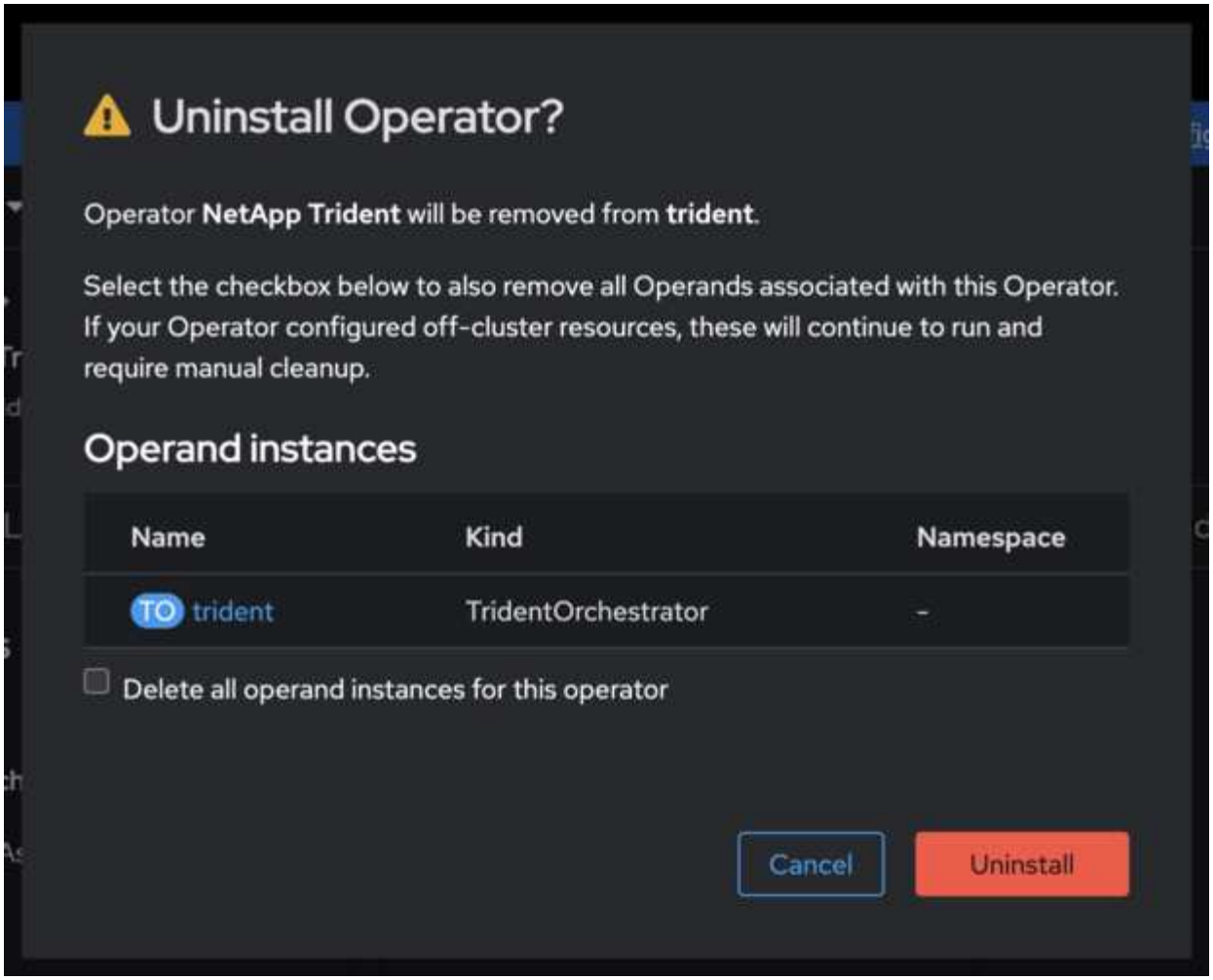
Passez de l'opérateur communautaire

Étapes

1. Utilisez la console OpenShift pour accéder à OperatorHub.



2. Trouvez l'opérateur de la communauté NetApp Trident.



Ne sélectionnez pas **Supprimer toutes les instances d'opérande de cet opérateur**.

3. Cliquez sur **Désinstaller**.
4. Une fois la désinstallation terminée, passez à Installez l'opérateur certifié OpenShift.

Passer d'une installation opérateur basée sur Helm

Étapes

1. Répertoriez la version Helm de votre installation Trident :

```
helm ls -n trident
```

2. Désinstallez la version Helm :

```
helm uninstall <release-name> -n trident
```

3. Une fois la désinstallation terminée, passez à Installez l'opérateur certifié OpenShift.

Passer d'un opérateur déployé manuellement

Si vous avez installé Trident en déployant manuellement l'opérateur à l'aide d'un `bundle.yaml` du package d'installation, supprimez-le en supprimant le même manifeste.

Étapes

1. Supprimez le déploiement de l'opérateur à l'aide du manifeste du bundle :

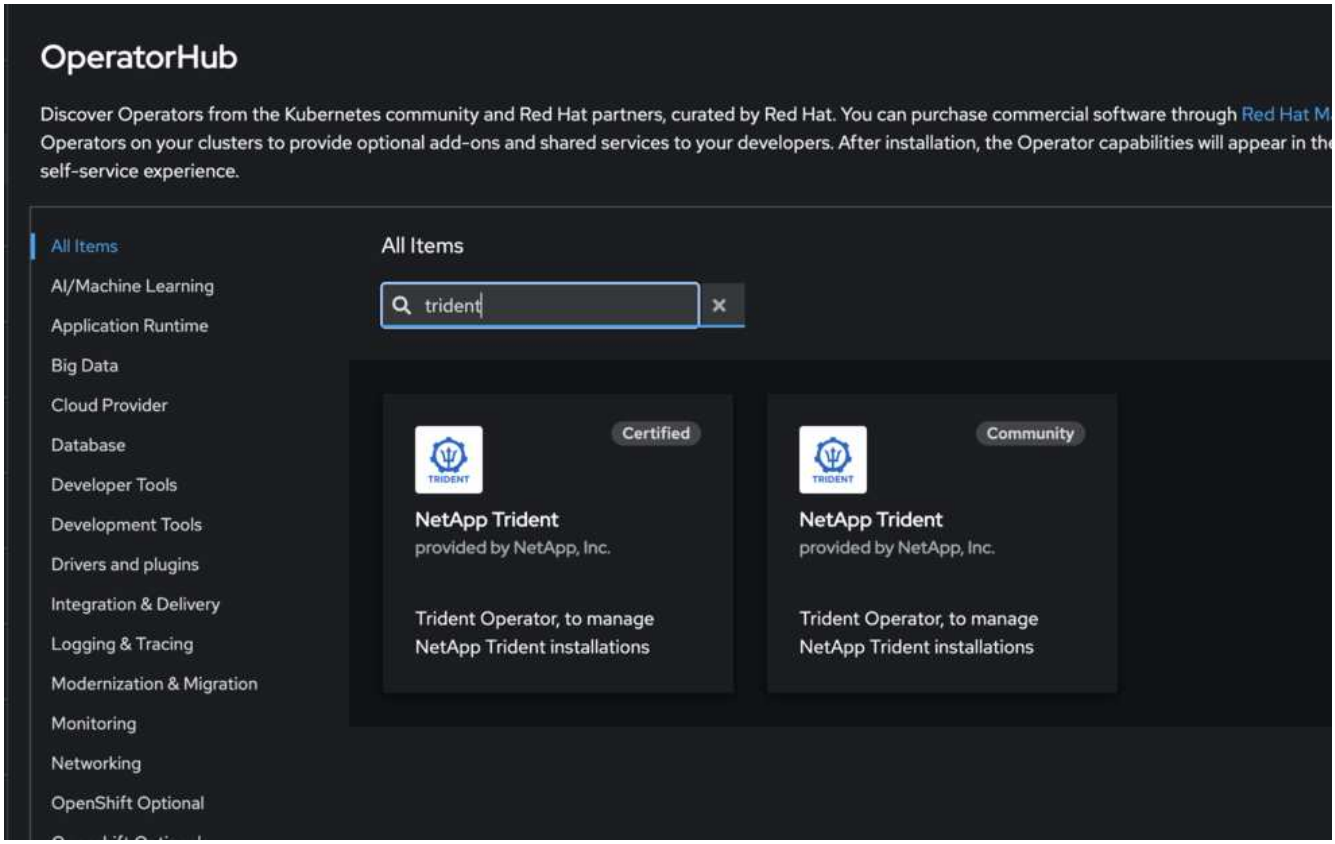
```
kubectl delete -f deploy/bundle.yaml -n trident
```

2. Une fois la désinstallation terminée, passez à Installez l'opérateur certifié OpenShift.

Installez l'opérateur certifié OpenShift

Étapes

1. Accédez à Red Hat OperatorHub.
2. Recherchez et sélectionnez l'opérateur NetApp Trident.



3. Suivre les instructions à l'écran pour installer l'opérateur.

Vérification

- Vérifiez le pupitre de commande de l'OperatorHub pour vous assurer que le nouvel opérateur certifié a été installé correctement.

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 outils installés par défaut sont installés dans les versions récentes de Red Hat Enterprise Linux CoreOS (RHCOS).

Outils NFS

"Installez les outils NFS"si vous utilisez : `ontap-nas` , `ontap-nas-economy` , `ontap-nas-flexgroup` , ou `azure-netapp-files` .

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).

> NetApp recommande ONTAP 9.12 ou version ultérieure pour NVMe/TCP.

Outils SCSI sur FC

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

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

Points à prendre en compte : * SCSI sur FC est pris en charge sur les environnements OpenShift et KubeVirt. * SCSI sur FC n'est pas pris en charge sur Docker. * L'auto-rétablissement iSCSI ne s'applique pas au SCSI sur FC.

Outils SMB

"Préparez-vous au provisionnement des volumes SMB" si vous utilisez : `ontap-nas` provisionner des volumes SMB.

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
```

- Lors de l'utilisation de nœuds worker exécutant RHEL/Red Hat Enterprise Linux CoreOS (RHCOS) avec des PV iSCSI, spécifiez la `discard` mountOption dans la classe de stockage pour effectuer la récupération d'espace en ligne. Reportez-vous à la "Documentation Red Hat".
- Assurez-vous d'avoir mis à niveau vers la dernière version du `multipath-tools` .

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. 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
```

> Assurez-vous que `/etc/multipath.conf` contient `find_multipaths no` moins de `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
```

> Assurez-vous que `/etc/multipath.conf` contient `find_multipaths no` moins de `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-100.2602.0.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.

Volumes SCSI sur FC

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

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"

Installez les outils FC

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

- Lors de l'utilisation de nœuds worker exécutant RHEL/Red Hat Enterprise Linux CoreOS (RHCOS) avec FC PVS, spécifiez la `discard` mountOption dans la classe de stockage pour effectuer la récupération d'espace en ligne. Reportez-vous à la "Documentation Red Hat".

RHEL 8+

1. Installez les packages système suivants :

```
sudo yum install -y lsscsi device-mapper-multipath
```

2. Activer les chemins d'accès multiples :

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

> Assurez-vous que `/etc/multipath.conf` contient `find_multipaths no` moins de `defaults`.

3. Assurez-vous que `multipathd` est en cours d'exécution :

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

Ubuntu

1. Installez les packages système suivants :

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

2. 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
```

> Assurez-vous que `/etc/multipath.conf` contient `find_multipaths no` moins de `defaults`.

3. Assurez-vous que `multipath-tools` est activé et en cours d'exécution :

```
sudo systemctl status multipath-tools
```

Préparez-vous au provisionnement des volumes SMB

Vous pouvez provisionner des volumes SMB à l'aide de `ontap-nas` conducteurs.

> Vous devez configurer les protocoles NFS et SMB/CIFS au SVM pour créer un `ontap-nas-economy` volume SMB pour les clusters 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.**	« »

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"
- "Configurez un système back-end Google Cloud NetApp volumes"
- "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

Utilisez Azure NetApp Files comme backend pour Trident. Ce backend prend en charge les volumes NFS et SMB. Trident prend en charge les identités managées et l'identité de charge de travail pour les clusters Azure Kubernetes Service (AKS).

Environnements cloud Azure pris en charge

Trident prend en charge les backends Azure NetApp Files dans plusieurs environnements cloud Azure.

Les clouds Azure pris en charge incluent :

- Azure Commercial
- Azure Government (Azure Government / MAG)

Lorsque vous déployez Trident ou configurez un backend Azure NetApp Files, assurez-vous qu'Azure Resource Manager et les points de terminaison d'authentification correspondent à votre environnement cloud.

Vérifiez la prise en charge du pilote Azure NetApp Files

Trident fournit le pilote de stockage Azure NetApp Files suivant.

Les modes d'accès pris en charge incluent *ReadWriteOnce* (RWO), *ReadOnlyMany* (ROX), *ReadWriteMany* (RWX) et *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 à examiner

- Azure NetApp Files ne prend pas en charge les volumes inférieurs à 50 Gio. Trident crée un volume de 50 Gio lorsqu'un volume plus petit est demandé.
- Trident prend en charge les volumes SMB montés sur les pods s'exécutant sur les nœuds Windows uniquement.
- Les déploiements Azure NetApp Files dans les clouds Azure non commerciaux nécessitent des points de terminaison Azure Resource Manager et d'authentification spécifiques au cloud. Assurez-vous que Trident et toute configuration backend utilisent les points de terminaison appropriés pour votre environnement cloud Azure.

Utilisez des identités gérées pour AKS

Trident prend en charge "identités gérées" pour les clusters AKS.

Si vous utilisez `tridentctl` pour créer ou gérer des backends Azure NetApp Files, assurez-vous qu'il est configuré pour le bon environnement cloud Azure.

Pour utiliser les identités gérées, vous devez avoir :

- Cluster Kubernetes déployé à l'aide d'AKS
- Identités gérées configurées sur le cluster Kubernetes AKS
- Trident installé avec `cloudProvider` défini sur `"Azure"`

Opérateur Trident

Modifier `tridentorchestrator_cr.yaml` et définir `cloudProvider` sur `"Azure"`.

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

Gouvernail

L'exemple suivant installe Trident et configure `cloudProvider` en utilisant la variable d'environnement `$CP` :

```
helm install trident trident-operator-100.2602.0.tgz --create-namespace
--namespace <trident-namespace> --set cloudProvider=$CP
```

<code>tridentctl</code>

L'exemple suivant installe Trident et définit le `cloud-provider` indicateur sur `Azure` :

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

Utiliser l'identité de charge de travail pour AKS

L'identité de charge de travail permet aux pods Kubernetes d'accéder aux ressources Azure en s'authentifiant en tant qu'identité de charge de travail.

Si vous utilisez `tridentctl` pour créer ou gérer des backends Azure NetApp Files, assurez-vous qu'il est configuré pour le bon environnement cloud Azure.

Pour utiliser l'identité de charge de travail, vous devez avoir :

- 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 avec `cloudProvider` défini sur `"Azure"` et `cloudIdentity` défini sur la valeur d'identité de la charge de travail

Opérateur Trident

Modifier `tridentorchestrator_cr.yaml` et définir `cloudProvider` sur `"Azure"`. Définir `cloudIdentity` sur `azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxx`.

```
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' # Edit
```

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 configure `cloudProvider` en utilisant `$CP` et configure `cloudIdentity` en utilisant `$CI` :

```
helm install trident trident-operator-100.6.0.tgz --set cloudProvider=$CP --set cloudIdentity="$CI"
```

<code>tridentctl</code>

Définissez les valeurs pour les indicateurs **fournisseur de cloud** et **identité cloud** 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 `cloud-provider` 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.

Environnements cloud Azure pris en charge

Trident prend en charge les backends Azure NetApp Files dans plusieurs environnements cloud Azure.

Les clouds Azure pris en charge incluent :

- Azure Commercial
- Azure Government (Azure Government / MAG)

Lors de la préparation de votre environnement, assurez-vous que votre abonnement Azure, votre configuration d'identité et vos ressources Azure NetApp Files sont créés dans l'environnement cloud Azure approprié.

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 nécessaire pour configurer Azure NetApp Files et créer un volume NFS. Consultez "Azure : configurez Azure NetApp Files et créez un volume NFS".

Pour configurer et utiliser un "Azure NetApp Files" backend, 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.
- Les déploiements Azure NetApp Files dans les clouds Azure non commerciaux nécessitent des points de terminaison Azure Resource Manager et d'authentification spécifiques au cloud. Assurez-vous que Trident et toute configuration backend utilisent les points de terminaison appropriés pour votre environnement cloud Azure.

- 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 backend (sous-réseau, réseau virtuel, niveau de service et emplacement) pour créer des volumes Azure NetApp Files sur des pools de capacité disponibles dans l'emplacement demandé et correspondant au niveau de service et au sous-réseau demandés.

Les systèmes Azure NetApp Files back-end proposent ces options de configuration.

Paramètre	Description	Valeur par défaut
`version`	Version de la configuration du backend.	Toujours 1
`storageDriverName`	Nom du pilote de stockage	« azure-netapp-files »
`backendName`	Nom personnalisé pour le backend 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)
`location`	Nom de l'emplacement Azure dans lequel les nouveaux volumes seront créés Facultatif lorsque les identités gérées sont activées sur un cluster AKS.	

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

Paramètre	Description	Valeur par défaut
`nasType`	Configurez la création de volumes NFS ou SMB. Les options sont `nfs`, `smb` ou nul. La valeur null par défaut sur les volumes NFS.	`nfs`
`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 .	
`qosType`	Indique le type de QoS : Auto ou Manuel.	Automatique
`maxThroughput`	Définit le débit maximal autorisé en Mio/s. Prise en charge uniquement pour les pools de capacité QoS manuels.	`4 MiB/sec`

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

Envisagez les environnements cloud Azure (26.02)

À partir de la version 26.02, Trident prend en charge la création et la gestion de backends Azure NetApp Files dans plusieurs environnements cloud Azure.

Les clouds Azure pris en charge incluent :

- Azure Commercial
- Azure Government (Azure Government / MAG)

Lorsque vous déployez Trident ou créez un backend Azure NetApp Files, assurez-vous que Azure Resource Manager et les points de terminaison d'authentification correspondent à votre environnement cloud. Si les points de terminaison ne correspondent pas, `tridentctl` ne peut pas s'authentifier et la création du backend échoue.

Autorisations et ressources requises

Si vous recevez l'erreur « Aucun pool de capacité trouvé » lors de la création d'un PVC, il est probable que l'enregistrement de votre application ne dispose pas des autorisations et ressources requises (sous-réseau, réseau virtuel, pool de capacité) associées. Si le débogage est activé, Trident enregistre les ressources Azure détectées lors de la création du backend. Vérifiez qu'un rôle approprié est utilisé.

Les valeurs pour `resourceGroups`, `netappAccounts`, `capacityPools`, `virtualNetwork` et `subnet` peuvent être spécifiées à l'aide de noms courts ou de noms complets. Les noms complets sont recommandés dans la plupart des situations, car les noms courts peuvent correspondre à plusieurs ressources portant le même nom.

Si le réseau virtuel est situé dans un groupe de ressources différent de celui du compte de stockage Azure NetApp Files (ANF), spécifiez le groupe de ressources du réseau virtuel lors de la configuration de la liste des groupes de ressources pour le backend.

Les `resourceGroups`, `netappAccounts`, et `capacityPools` valeurs sont des filtres qui limitent l'ensemble des ressources découvertes à celles disponibles pour ce système de stockage et peuvent être spécifiées dans n'importe quelle combinaison. Les noms pleinement qualifiés suivent ce format :

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 des volumes par défaut en spécifiant les options suivantes dans une section spéciale du fichier de configuration. Consultez Exemples de configurations pour plus de détails.

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`	Accès au `.snapshot` répertoire	`true`, `false` (Défini explicitement).
`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 présentent des configurations de base qui laissent la plupart des paramètres par défaut. C'est la manière la plus simple de définir un backend.

Configuration minimale

Il s'agit de la configuration minimale absolue du backend. Avec cette configuration, Trident découvre tous vos comptes NetApp, pools de capacité et sous-réseaux délégués à Azure NetApp Files dans l'emplacement configuré, et place les nouveaux volumes aléatoirement sur l'un de ces pools et sous-réseaux. Parce que `nasType` est omis, la `nfs` configuration par défaut s'applique et le backend provisionnera des 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:
    - resource-group-1/netapp-account-1/ultra-pool
  resourceGroups:
    - resource-group-1
  netappAccounts:
    - resource-group-1/netapp-account-1
  virtualNetwork: resource-group-1/eastus-prod-vnet
  subnet: resource-group-1/eastus-prod-vnet/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 backend place les volumes dans l'emplacement `eastus` d'Azure dans un `Ultra` pool de capacité. Trident détecte automatiquement tous les sous-réseaux délégués à Azure NetApp Files dans cet emplacement et place un nouveau volume sur l'un d'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
```

Exemple de backend avec pools de capacité QoS manuels

Cette configuration backend place les volumes dans Azure. `eastus` emplacement avec pools de capacité QoS manuels.

```
---
version: 1
storageDriverName: azure-netapp-files
backendName: anf1
location: eastus
labels:
  clusterName: test-cluster-1
  cloud: anf
  nasType: nfs
defaults:
  qosType: Manual
storage:
  - serviceLevel: Ultra
    labels:
      performance: gold
    defaults:
      maxThroughput: 10
  - serviceLevel: Premium
    labels:
      performance: silver
    defaults:
      maxThroughput: 5
  - serviceLevel: Standard
    labels:
      performance: bronze
    defaults:
      maxThroughput: 3
```

Configuration avancée

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

```
---
version: 1
storageDriverName: azure-netapp-files
subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
tenantID: 68e4f836-edc1-fake-bff9-b2d865ee56cf
clientID: dd043f63-bf8e-fake-8076-8de91e5713aa
clientSecret: SECRET
location: eastus
serviceLevel: Ultra
capacityPools:
  - application-group-1/account-1/ultra-1
  - application-group-1/account-1/ultra-2
virtualNetwork: application-group-1/eastus-prod-vnet
subnet: application-group-1/eastus-prod-vnet/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 backend définit plusieurs pools de stockage dans un seul fichier. Ceci est utile lorsque vous disposez de plusieurs pools de capacité prenant en charge différents niveaux de service et que vous souhaitez créer des classes de stockage dans Kubernetes pour les représenter. Des étiquettes de pool virtuel 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:
      - application-group-1/netapp-account-1/ultra-1
      - application-group-1/netapp-account-1/ultra-2
    networkFeatures: Standard
  - labels:
      performance: silver
    serviceLevel: Premium
    capacityPools:
      - application-group-1/netapp-account-1/premium-1
  - labels:
      performance: bronze
    serviceLevel: Standard
    capacityPools:
      - application-group-1/netapp-account-1/standard-1
      - application-group-1/netapp-account-1/standard-2
```

Configuration des topologies prises en charge

Trident facilite le provisionnement des volumes pour les charges de travail en fonction des régions et des zones de disponibilité. Le `supportedTopologies` bloc dans cette configuration backend est utilisé pour fournir une liste de régions et de zones par backend. Les valeurs de région et de zone spécifiées ici doivent correspondre aux valeurs de région et de zone des étiquettes sur 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 backend, Trident crée les volumes dans la région et la zone mentionnées. Pour plus d'informations, consultez "Utiliser la topologie CSI".

```
---
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

En utilisant `parameter.selector`, vous pouvez spécifier pour chaque `StorageClass` le pool virtuel utilisé pour héberger un volume. Le volume aura les aspects définis dans le pool choisi.

```
---
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

En utilisant `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 prenant en charge les volumes SMB. `nasType: nfs` ou `nasType: null`filtres pour les pools NFS.

Créer le backend

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

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

Si vous utilisez un cloud Azure non commercial, assurez-vous que `tridentctl` est configuré pour utiliser Azure Resource Manager et les points de terminaison d'authentification de votre environnement cloud. Si la création du backend échoue, vérifiez la configuration de votre backend et consultez les journaux pour en déterminer la cause :

```
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

Configurer Google Cloud NetApp Volumes

Vous pouvez configurer Google Cloud NetApp Volumes comme backend pour que Trident puisse provisionner du stockage pour les charges de travail Kubernetes.

Présentation

Trident prend en charge Google Cloud NetApp Volumes pour les charges de travail NAS (NFS et SMB) et bloc (iSCSI).

- Les charges de travail NAS utilisent le `google-cloud-netapp-volumes` backend
- Les charges de travail par blocs (iSCSI) utilisent le `google-cloud-netapp-volumes-san` backend

Les volumes NAS offrent un stockage de fichiers et sont accessibles via les protocoles NFS ou SMB. Ces volumes prennent en charge l'accès partagé entre plusieurs pods ou nœuds.

Les volumes de blocs offrent un stockage bloc et sont accessibles comme des périphériques iSCSI connectés aux nœuds Kubernetes. Ces volumes sont utilisés lorsque les applications nécessitent un accès au niveau bloc.

Ceci s'applique aux environnements suivants :

- Trident 26.02 et versions ultérieures
- Google Kubernetes Engine (GKE) ou Red Hat OpenShift
- Pools de stockage Google Cloud NetApp Volumes

Pour configurer le stockage bloc (iSCSI), voir "Configurer le stockage bloc (iSCSI)".

Préparez la configuration

L'identité cloud permet aux charges de travail Kubernetes d'accéder aux ressources Google Cloud en s'authentifiant en tant qu'identité de charge de travail au lieu d'utiliser des informations d'identification statiques.

Pour utiliser l'identité cloud avec Google Cloud NetApp Volumes, vous devez disposer des éléments suivants :

- Un cluster Kubernetes déployé à l'aide de Google Kubernetes Engine (GKE)
- L'identité de la charge de travail est activée sur le cluster GKE et le serveur de métadonnées est activé sur les pools de nœuds
- Un compte de service Google Cloud avec le rôle d'administrateur Google Cloud NetApp Volumes (`roles/netapp.admin`) ou un rôle personnalisé équivalent
- Trident installé avec le fournisseur de cloud défini sur `GCP` et l'annotation d'identité cloud configurée

Opérateur Trident

Pour installer Trident à l'aide de l'opérateur Trident, modifiez `tridentorchestrator_cr.yaml` :

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

Gouvernail

Configurez le fournisseur de cloud et l'identité cloud lors de l'installation de Trident avec Helm :

```
helm install trident trident-operator-100.6.0.tgz \
  --set cloudProvider=GCP \
  --set cloudIdentity="iam.gke.io/gcp-service-account: cloudvolumes-
admin-sa@mygcpproject.iam.gserviceaccount.com"
```

tridentctl

Installez Trident en spécifiant le fournisseur de cloud et l'identité cloud :

```
tridentctl install \
  --cloud-provider=GCP \
  --cloud-identity="iam.gke.io/gcp-service-account: cloudvolumes-admin-
sa@mygcpproject.iam.gserviceaccount.com" \
  -n trident
```

Configurer le stockage NAS

Pour les pools de stockage UNIFIED de Google Cloud NetApp Volumes, Trident applique des règles de nommage et de validation spécifiques à UNIFIED lors des opérations sur les volumes.

Lors de la localisation d'un volume, Trident peut évaluer plusieurs variantes de nom de volume compatibles (par exemple, les formats avec tiret et trait de soulignement) afin d'améliorer la fiabilité de l'importation et de la découverte.

Détails du pilote

Trident fournit le `google-cloud-netapp-volumes` driver pour provisionner le stockage NAS à partir de Google Cloud NetApp Volumes.

Le pilote prend en charge les modes d'accès suivants :

- ReadWriteOnce (RWO)
- ReadOnlyMany (ROX)
- ReadWriteMany (RWX)
- ReadWriteOncePod (RWOP)

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

Configurer un backend NAS Trident

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: gcnv-nas
  namespace: trident
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: "<project-number>"
  location: "<region>"
  sdkTimeout: "600"
  storage:
  - labels:
      cloud: gcp
    network: "<vpc-network>"
```

Provisionner des volumes NAS

Les volumes NAS sont provisionnés à l'aide du `google-cloud-netapp-volumes` backend et prennent en charge les protocoles NFS et SMB.

StorageClass for les volumes NFS

Pour provisionner des volumes NFS, définissez `nasType` sur `nfs`.

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

StorageClass for les volumes SMB

Pour provisionner des volumes SMB, configurez `nasType` sur `smb` et fournissez les informations d'identification.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gcnv-smb
provisioner: csi.trident.netapp.io
parameters:
  backendType: "google-cloud-netapp-volumes"
  trident.netapp.io/nasType: "smb"
  csi.storage.k8s.io/node-stage-secret-name: "smbcreds"
  csi.storage.k8s.io/node-stage-secret-namespace: "default"
allowVolumeExpansion: true
```

Exemple de PersistentVolumeClaim (RWX)

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

Exemple de PersistentVolumeClaim (RWO)

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: gcnv-nas-rwo
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 100Gi
  storageClassName: gcnv-nfs
```

Les volumes NAS utilisent `volumeMode: Filesystem`.

Configurer Google Cloud NetApp Volumes pour les charges de travail SAN

Vous pouvez configurer Trident pour provisionner des volumes de stockage bloc à l'aide du protocole iSCSI depuis Google Cloud NetApp Volumes. Les volumes SAN sont provisionnés à partir de pools de stockage Flex Unified en utilisant le `google-cloud-netapp-volumes-san` storage driver.

Ce pilote est dédié aux charges de travail bloc et ne prend pas en charge les protocoles NAS.

Le `google-cloud-netapp-volumes-san` backend est requis pour provisionner des volumes de blocs iSCSI. Le `google-cloud-netapp-volumes` backend prend uniquement en charge les protocoles NAS et ne peut pas être utilisé pour les charges de travail SAN.

Présentation

Trident prend en charge les charges de travail Google Cloud NetApp Volumes SAN (iSCSI) à l'aide du `google-cloud-netapp-volumes-san` driver.

Les volumes SAN sont provisionnés à partir de pools de stockage Flex Unified et présentés aux nœuds Kubernetes en tant que périphériques de blocs iSCSI.

Ceci s'applique aux environnements suivants :

- Trident 26.02 et versions ultérieures
- Google Kubernetes Engine (GKE) ou Red Hat OpenShift
- Google Cloud NetApp Volumes Flex Pools de stockage unifiés
- Charges de travail basées sur iSCSI

Pools de stockage Flex Unified

Les pools de stockage Flex Unified fournissent du stockage bloc utilisant le protocole iSCSI et sont nécessaires au provisionnement SAN :

- Les pools régionaux Flex Unified sont pris en charge.
- Les pools ZONAL Flex Unified sont pris en charge à partir de Trident 26.02.1.
- Seul le niveau de service **Flex** est pris en charge pour les charges de travail SAN.

Configurer un backend SAN Trident

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: gcnv-san
  namespace: trident
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes-san
  projectNumber: "<project-number>"
  location: "<region>"
  sdkTimeout: "600"
  storage:
  - labels:
      cloud: gcp
      performance: flex
    network: "<vpc-network>"
    serviceLevel: Flex
```

Créer un StorageClass

Après avoir configuré le backend SAN, créez un StorageClass qui référence le `google-cloud-netapp-volumes-san` driver.

Le type de système de fichiers est défini dans le StorageClass, pas dans le backend.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gcnv-san
provisioner: csi.trident.netapp.io
parameters:
  backendType: "google-cloud-netapp-volumes-san"
  fsType: "ext4"
allowVolumeExpansion: true
```

Types de systèmes de fichiers pris en charge :

- `ext4` (défaut)
- `ext3`
- `xfs`

> Le pilote SAN ne prend en charge que le niveau de service Flex et n'utilise pas les paramètres backend spécifiques au NAS tels que `exportRule`, `unixPermissions`, `nasType`, `snapshotDir`, `nfsMountOptions` ou les paramètres liés à la hiérarchisation.

Approvisionner des volumes de blocs

ReadWriteOnce (RWO)

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: gcnv-san-rwo
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 100Gi
  storageClassName: gcnv-san
```

ReadWriteOncePod (RWOP)

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: gcnv-san-rwop
spec:
  accessModes:
    - ReadWriteOncePod
  resources:
    requests:
      storage: 100Gi
  storageClassName: gcnv-san
```

ReadOnlyMany (ROX)

Une méthode courante pour ROX consiste à cloner un volume ReadWriteOnce existant et à monter le clone en lecture seule.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: gcnv-san-rox
spec:
  accessModes:
    - ReadOnlyMany
  resources:
    requests:
      storage: 100Gi
  storageClassName: gcnv-san
  dataSource:
    kind: PersistentVolumeClaim
    name: gcnv-san-rwo
```

ReadWriteMany (RWX) — bloc brut uniquement

ReadWriteMany n'est pris en charge que lorsque `volumeMode: Block`.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: gcnv-san-raw-rwx
spec:
  accessModes:
    - ReadWriteMany
  volumeMode: Block
  resources:
    requests:
      storage: 100Gi
  storageClassName: gcnv-san
```

Comportement du volume de bloc

Les volumes de blocs sont provisionnés en tant que LUN iSCSI et présentés aux nœuds Kubernetes en tant que périphériques de blocs.

Volumes de blocs :

- Utilisez le protocole iSCSI
- Prise en charge du système de fichiers et de la présentation des blocs bruts
- Sont rattachés et gérés par Trident
- Prise en charge de plusieurs modes d'accès Kubernetes

Modes d’accès

Les volumes de blocs provisionnés par Trident prennent en charge les modes d’accès suivants :

- `ReadWriteOnce` (RWO)
- `ReadOnlyMany` (ROX)
- `ReadWriteOncePod` (RWOP)
- `ReadWriteMany` (RWX), pris en charge uniquement lorsque `volumeMode: Block`

Comportement de volumeMode

Le `volumeMode`champ contrôle la façon dont un volume de bloc est exposé :

- `Filesystem` Trident formate et monte le volume.
- `Block` Trident attache le périphérique et l’expose comme un périphérique de bloc brut.

Opérations prises en charge

Volumes de blocs provisionnés à l’aide du pilote `google-cloud-netapp-volumes-san` prennent en charge :

- Créer
- Supprimer
- Cloner
- Snapshot
- Redimensionner
- Importer

Comportement de surprovisionnement supplémentaire de GiB

Google Cloud NetApp Volumes block volumes incluent des métadonnées internes. Cette surcharge réduit la taille du périphérique visible par le noyau par rapport à la capacité provisionnée.

Les tests montrent :

- Environ 300 Kio de surcharge lors de la création initiale
- Jusqu’à environ 107 MiB de surcharge après un redimensionnement

Étant donné que Google Cloud NetApp Volumes n’accepte que des allocations de Gio entiers, Trident garantit que la taille utilisable du périphérique respecte ou dépasse toujours la demande PVC en :

- Arrondi de la taille demandée au Gio entier supérieur
- Ajout d’un tampon supplémentaire de 1 Gio

Exemple :

- Requête PVC : 100 Gio
- Taille provisionnée dans Google Cloud NetApp Volumes : 101 GiB

- Espace utilisable visible par l'application : au moins 100 GiB

Exemples de pods

Volume de bloc monté sur le système de fichiers (RWO)

```
apiVersion: v1
kind: Pod
metadata:
  name: app-rwo
spec:
  containers:
  - name: app
    image: ubuntu:22.04
    command: ["sleep", "infinity"]
    volumeMounts:
    - name: data
      mountPath: /mnt/data
  volumes:
  - name: data
    persistentVolumeClaim:
      claimName: gcnv-san-rwo
```

Périphérique de bloc brut (RWX)

```
apiVersion: v1
kind: Pod
metadata:
  name: app-raw-rwx
spec:
  containers:
  - name: app
    image: ubuntu:22.04
    command: ["sleep", "infinity"]
    volumeDevices:
    - name: data
      devicePath: /dev/xda
  volumes:
  - name: data
    persistentVolumeClaim:
      claimName: gcnv-san-raw-rwx
```

Comportement d'attachement et de montage

Pour les volumes SAN provisionnés à partir de Google Cloud NetApp Volumes :

- Trident crée un Logical Unit Number (LUN) dans un pool de stockage Flex Unified.
- Lors de la publication, Trident associe le LUN à un groupe d'hôtes par nœud.
- Lors de la mise en scène des nœuds, Trident :
  - Se connecte à la cible iSCSI
  - Découvre le LUN
  - Configure le multipath
- Si `volumeMode: Filesystem`Trident le nécessite, Trident formate le périphérique et le monte.
- Si `volumeMode: Block` Trident attache le périphérique et l'expose directement au pod sans le formater ni le monter.

> Les volumes de blocs SAN ne prennent pas en charge le verrouillage distribué ni la coordination des écritures. Lorsqu'un volume de blocs est accédé par plusieurs nœuds (ReadWriteMany avec `volumeMode: Block`), l'application ou le système de fichiers doit gérer la concurrence.

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 ou SMB

Si vous utilisez Google Cloud NetApp Volumes pour la première fois ou dans un nouvel emplacement, une configuration initiale est nécessaire pour configurer Google Cloud NetApp Volumes et créer un volume NFS ou SMB. Consultez "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 (`roles/netapp.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 back-end 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 ».
`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 `netapp.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`.	
`labels`	Ensemble d'étiquettes arbitraires au format JSON à appliquer aux volumes	« »
`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

Paramètre	Description	Valeur par défaut
`nasType`	Configurez la création de volumes NFS ou SMB. Les options sont `nfs`, `smb` ou nul. La valeur null par défaut sur les volumes NFS.	`nfs`
`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`, `false` (Le comportement par défaut peut varier. À définir explicitement) « 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 débutez avec Google Cloud NetApp Volumes et que vous faites des essais, mais en pratique, vous pourriez avoir besoin de définir une portée supplémentaire pour les volumes que vous provisionnez.

Remplacez `<id_value>` et `<key_value>` par vos identifiants de compte de service.

```
---
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-gcnv-secret
type: Opaque
stringData:
  private_key_id: "<id_value>"
  private_key: |
    -----BEGIN PRIVATE KEY-----
    <key_value>
    -----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
  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 pour les volumes SMB

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-gcnv1
  namespace: trident
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: "123456789"
  location: asia-east1
  serviceLevel: flex
  nasType: smb
  apiKey:
    type: service_account
    project_id: cloud-native-data
    client_email: trident-sample@cloud-native-
data.iam.gserviceaccount.com
    client_id: "123456789737813416734"
    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/trident-
sample%40cloud-native-data.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: "<id_value>"
  private_key: |
    -----BEGIN PRIVATE KEY-----
    <key_value>
    -----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: "<id_value>"
  private_key: |
    -----BEGIN PRIVATE KEY-----
    <key_value>
    -----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

Lors de l'installation de Trident avec une identité cloud GKE, les backends s'authentifient en tant qu'identité de charge de travail et omettent le `apiKey` bloc. L'identité cloud est configurée de la même manière pour les deux pilotes Google Cloud NetApp Volumes. La seule différence concerne la valeur `storageDriverName` : utilisez `google-cloud-netapp-volumes` pour les volumes NAS (NFS ou SMB) et `google-cloud-netapp-volumes-san` pour les volumes SAN (bloc iSCSI).

> La prise en charge de l'identité cloud pour le `google-cloud-netapp-volumes-san` driver nécessite Trident 26.06 ou une version ultérieure. Les volumes SAN sont provisionnés à partir de pools de stockage Flex Unified, donc définissez `serviceLevel` sur `flex`.

Pilote NAS (<code>google-cloud-netapp-volumes</code>)

```
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
```

Pilote SAN (<code>google-cloud-netapp-volumes-san</code>)

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-gcp-gcnv-san
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes-san
  projectNumber: '012345678901'
  network: gcnv-network
  location: us-west2
  serviceLevel: flex
  storagePool: pool-flex1
```

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.

Définitions des classes 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 .

Exemples de définitions pour les volumes SMB

A 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. N'importe quel utilisateur/mot de passe Active Directory avec des autorisations n'importe quel/aucune peut être utilisé pour le secret d'étape du nœud.

Configuration de base sur l'espace de noms par défaut

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

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 la hiérarchisation automatique pour Google Cloud NetApp Volumes

La hiérarchisation automatique est configurée via les paramètres du backend Trident et les annotations PersistentVolumeClaim lors du provisionnement des volumes. Vous pouvez configurer la hiérarchisation automatique pour Google Cloud NetApp Volumes à l'aide de Trident.

Présentation

La hiérarchisation automatique permet à Trident de provisionner des volumes qui déplacent automatiquement les données inactives d'un niveau de performance vers un niveau de capacité. Cela réduit le coût de stockage tout en préservant les performances pour les données fréquemment consultées.

Trident applique les paramètres de hiérarchisation automatique uniquement lors de la création du volume. Les modifications après le provisionnement ne sont pas prises en charge dans Trident 26.02.

Concepts

hiérarchisation automatique

Le système de hiérarchisation automatique déplace les données rarement consultées d'un niveau de performance vers un niveau de capacité en fonction des modèles d'accès. Le déplacement des données s'effectue de manière asynchrone et n'est pas immédiat.

politique de hiérarchisation

La politique de hiérarchisation détermine si la hiérarchisation automatique est activée pour un volume.

Les politiques suivantes sont prises en charge : * `auto` : Active la hiérarchisation automatique en fonction des modèles d'accès * `none` : Désactive la hiérarchisation automatique

Jours de refroidissement

Les jours de refroidissement spécifient le nombre minimal de jours pendant lesquels un bloc de données doit rester inactif avant de pouvoir être classé par niveau. Les jours de refroidissement s'appliquent uniquement lorsque la politique de classement par niveau est définie sur `auto`.

Modèle de configuration

Étendues de configuration

La hiérarchisation automatique peut être configurée à plusieurs niveaux :

- **Portée du pool de stockage** S'applique à tous les volumes provisionnés à partir du pool.
- **Portée du volume** S'applique à un seul volume par le biais des annotations PersistentVolumeClaim.

Trident détermine la configuration effective en fonction de l'endroit où chaque paramètre est défini.

Précédence de configuration

Lorsque le même paramètre est défini à plusieurs niveaux, Trident applique l'ordre de priorité suivant :

1. Annotations PersistentVolumeClaim
2. Configuration du backend Trident
3. Paramètres par défaut du storage pool

Les paramètres définis à un niveau de priorité supérieur remplacent les valeurs de niveau inférieur.

fonctionnalité prise en charge dans Trident 26.02

Trident 26.02 prend en charge les fonctionnalités de hiérarchisation automatique suivantes pour Google Cloud NetApp Volumes :

- Activation ou désactivation du auto-tiering lors de l'allocation des volumes
- Définition d'une politique de hiérarchisation dans la configuration du backend Trident
- Remplacement de la politique de hiérarchisation et du nombre de jours de refroidissement par volume à l'aide d'annotations PVC
- Configuration des jours de refroidissement pour les volumes avec auto-tiering activé

fonctionnalité non prise en charge dans Trident 26.02

Les opérations suivantes ne sont pas prises en charge :

- Modification des paramètres de hiérarchisation automatique après la création du volume
- Modification des politiques de hiérarchisation sur les volumes existants à l'aide des mises à jour Kubernetes
- Application des paramètres de hiérarchisation automatique en dehors des workflows de provisionnement gérés par Trident

Paramètres de configuration du backend

Les paramètres suivants contrôlent le comportement de hiérarchisation automatique lorsqu'ils sont définis dans la configuration du backend Trident :

Paramètre	Obligatoire	Description
tieringPolicy	Non	Politique de hiérarchisation des volumes (`auto` ou `none`)
tieringMinimumCoolingDays	Non	Nombre de jours d'inactivité avant le classement des données (plage : 2–183, valeur par défaut : 31)

Remplacements au niveau du volume à l'aide des annotations PersistentVolumeClaim

Annotations prises en charge

Les annotations PersistentVolumeClaim permettent de remplacer, volume par volume, les paramètres de hiérarchisation automatique.

Annotation	Description
`trident.netapp.io/tieringPolicy`	Remplace la politique de hiérarchisation pour le volume
`trident.netapp.io/tieringMinimumCoolingDays`	Remplace la valeur des jours de refroidissement pour le volume

Exemple : PersistentVolumeClaim avec des substitutions de hiérarchisation automatique

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: auto-tiering-pvc
  annotations:
    trident.netapp.io/tieringPolicy: auto
    trident.netapp.io/tieringMinimumCoolingDays: "45"
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: google-cloud-netapp-volumes-auto-tiering
  resources:
    requests:
      storage: 500Gi
```

Comportement et limitations

Comportement de provisionnement

- Les paramètres de hiérarchisation automatique sont évalués et appliqués uniquement lors de la création du volume.
- Trident ne réconcilie pas la configuration de hiérarchisation après le provisionnement.
- Les jours de refroidissement sont ignorés lorsque la politique de hiérarchisation est définie sur `none`.

Limitations de la plateforme

- La hiérarchisation automatique est prise en charge uniquement pour les volumes NAS (NFS et SMB).
- Les volumes de blocs (iSCSI) ne prennent pas en charge l'auto-tiering.
- Le pool de stockage Google Cloud NetApp Volumes doit avoir la hiérarchisation automatique activée dans Google Cloud.

Valeurs prises en charge

- Plage de valeurs valide `tieringMinimumCoolingDays`: 2 à 183
- Valeur par défaut : 31

Configurer les pools de stockage Google Cloud NetApp Volumes en mode ONTAP

À partir de Trident 26.06, vous pouvez configurer des backends existants `ontap-san` et `ontap-nas` pour provisionner du stockage sur Google Cloud NetApp Volumes (GCNV), dans des pools de stockage en mode ONTAP. Cette fonctionnalité est une Tech Preview.

REMARQUE	Cette fonctionnalité est une Tech Preview dans Trident 26.06. Les fonctionnalités Tech Preview ne sont pas prises en charge pour une utilisation en production. La fonctionnalité, les champs de configuration et la matrice prise en charge peuvent changer avant la disponibilité générale (GA).

Comment ça marche

Trident réutilise la logique de pilote existante `ontap-san` et `ontap-nas` et achemine les appels REST ONTAP via un point de terminaison proxy GCNV. Cette approche vous permet de provisionner et de gérer le stockage sur des clusters ONTAP hébergés sur GCNV tout en conservant les mêmes flux de travail opérationnels que pour un backend ONTAP direct. Le mode ONTAP n'introduit pas de nouveau pilote de stockage. Vous l'activez pour chaque backend en ajoutant un bloc de configuration `gcnv` à un backend `ontap-san` ou `ontap-nas`.

Le processus de provisionnement est le suivant :

`PersistentVolumeClaim` → Trident (`ontap-san` ou `ontap-nas`) → client REST ONTAP → proxy GCNV → cluster ONTAP dans le pool GCNV

Portée prise en charge et non prise en charge

Cette préversion technique prend en charge les éléments suivants :

- Pilotes : `ontap-san` (iSCSI) et `ontap-nas` (NFS ou SMB).
- Opérations de cycle de vie via le proxy : initialisation du backend, création et suppression de volumes, et flux REST ONTAP associés.
- Modèles d'authentification : Workload Identity Pool (WIP), clé de compte de service et repli Application Default Credentials (ADC).

Les éléments suivants ne font pas partie du champ d'application de cet aperçu technique :

- Les `ontap-san-economy`, `ontap-nas-economy` et `ontap-nas-flexgroup` pilotes.
- Le chemin de personnalité ASA r2.
- Solution de repli ZAPI. Le mode ONTAP utilise uniquement ONTAP REST.

Prérequis

Avant de configurer un backend en mode ONTAP, assurez-vous de disposer des éléments suivants :

- Trident 26.06 ou version ultérieure.
- Un pool de stockage GCNV en mode ONTAP dans votre projet et emplacement cibles.
- Autorisations GCP IAM pour les opérations de proxy, adaptées à votre environnement.
- Une configuration backend qui inclut `proxyURL`, `projectNumber`, `location` et `poolID`.
- Pour les flux de travail SAN iSCSI, les prérequis iSCSI côté nœud et multipath de votre plateforme. Consultez "Préparez le nœud de travail".

Configurer le backend

A `TridentBackendConfig` nécessite `credentials`. Pour le mode ONTAP, ce secret contient les identifiants du proxy GCP, et non ONTAP `managementLIF`/ `username`/ `password`. `svm` est facultatif : chaque pool GCNV en mode ONTAP possède actuellement une SVM, et Trident la déduit du pool lorsqu'il est omis.

Utilisez le même modèle de compte de service que GCNV natif : champs non sensibles dans `gcnv.apiKey`, `private_key` et `private_key_id` dans le secret. Voir "Exemples de configuration de Google Cloud NetApp Volumes".

backend ontap-san

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: gcnv-ontap-san
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-san
  backendName: gcnv-ontap-san
  credentials:
    name: gcnv-sa-secret
    type: secret
  gcnv:
    proxyURL: "https://netapp.googleapis.com"
    projectNumber: "<project-number>"
    location: "<region-or-zone>"
    poolID: "<pool-id>"
  svm: "<svm-name>"
```

backend ontap-nas

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: gcnv-ontap-nas
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-nas
  backendName: gcnv-ontap-nas
  credentials:
    name: gcnv-sa-secret
    type: secret
  gcnv:
    proxyURL: "https://netapp.googleapis.com"
    projectNumber: "<project-number>"
    location: "<region-or-zone>"
    poolID: "<pool-id>"
    apiKey:
      type: service_account
      project_id: "<project-id>"
      client_email: "<service-account-email>"
      client_id: "<client-id>"
      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: "<client-x509-cert-url>"
```

Secret

```
apiVersion: v1
kind: Secret
metadata:
  name: gcnv-sa-secret
  namespace: trident
type: Opaque
stringData:
  private_key_id: "<private-key-id>"
  private_key: |
    -----BEGIN PRIVATE KEY-----
    <private-key>
    -----END PRIVATE KEY-----
```

REMARQUE	N'incluez pas `private_key` ou `private_key_id` dans la `TridentBackendConfig` spécification : la CRD les rejette. Utilisez `ontap-san` avec le même `gcnv` et le même modèle d'identifiant pour le stockage bloc.

```
svm: "<svm-name>"
```

```
== Authentication

Trident resolves credentials for proxy access in the following order:

. Workload Identity Pool (WIP)
. Service account key
. Application Default Credentials (ADC)

For this Tech Preview, use the following practices:

* Prefer Workload Identity Pool where it is available.
* Do not embed raw private keys in version-controlled backend files. Store
credentials in a Kubernetes secret.
* Apply least-privilege IAM scoping to the service account.

== Validation and initialization behavior

Trident validates an ONTAP-mode backend during initialization and fails
fast rather than partially initializing.

Backend initialization fails if a required `gcnv` field is missing, the
driver is unsupported for ONTAP-mode, or credential resolution fails.

== Known limitations

* This feature is a Tech Preview. Behavior and the supported matrix can
change before GA.
* ZAPI fallback is not used in ONTAP-mode.
* Protocol and node-readiness requirements still apply. For example, iSCSI
workflows require the node-side prerequisites described in
xref:{relative_path}../trident-use/worker-node-prep.html[Prepare the
worker node].
* Existing ONTAP and GCNV environment constraints still apply.

== Upgrade and compatibility

ONTAP-mode is opt-in per backend through the `gcnv` configuration block:

* Existing non-GCNV ONTAP backends are unaffected.
```

```
* Mixed deployments that combine direct ONTAP backends and ONTAP-mode
proxy backends are supported through backend-level configuration. Each
backend is configured independently.
* To stop using ONTAP-mode, remove or replace the affected backends. No
global switch is required.

== What's next?

Apply secret before TBC; use `kubectl apply` and `-n trident` to match
examples.
```

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 »
`backendName`	Nom personnalisé ou système back-end de stockage	Adresse IP « SolidFire_ » + 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, "method":true}	nul

AVERTISSEMENT 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 SCSI sur FC	Bloc	RWO, ROX, RWX, RWOP	Pas de système de fichiers, périphérique de bloc brut
`ontap-san`	ISCSI SCSI sur FC	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`	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`
`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`

AVERTISSEMENT

- 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.
- NetApp ne recommande pas l'utilisation de l'autogrow FlexVol dans tous les pilotes ONTAP, sauf ONTAP-san. Pour contourner ce problème, Trident prend en charge l'utilisation de la réserve Snapshot et adapte les volumes FlexVol en conséquence.

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.

REMARQUE	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 nativement par NVMe
- Chemins d'accès multiples du mappeur de périphériques (DM)
- Cryptage LUKS

REMARQUE	NVMe est pris en charge uniquement avec les API REST ONTAP et n'est pas pris en charge avec ONTAPI (ZAPI).

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 backends ONTAP, Trident exige qu'au moins un agrégat soit attribué au SVM.

REMARQUE	"Systèmes ASA r2"diffèrent des autres systèmes ONTAP (ASA, AFF et FAS) dans la mise en œuvre de leur couche de stockage. Dans les systèmes ASA r2, on utilise des zones de disponibilité de stockage au lieu d'agrégats. Se référer à"c'est ça" Article de la base de connaissances sur la manière d'attribuer des agrégats aux SVM dans les systèmes ASA r2.

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.

AVERTISSEMENT	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

```yaml
---
version: 1
backendName: ExampleBackend
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_nfs
username: vsadmin
password: password
```

JSON

```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 les certificats

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

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

Un flux de travail type comprend les étapes suivantes.

Étapes

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

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

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

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

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

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

REMARQUE	Après avoir exécuté cette commande, ONTAP vous invite à saisir un certificat. Collez le contenu du `k8senv.pem` fichier généré à l'étape 1, puis appuyez sur `END` pour terminer l'installation.

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

REMARQUE	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 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
```

AVERTISSEMENT	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 ça marche

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.

Faire pivoter les informations d'identification et mettre à jour les backends

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.

AVERTISSEMENT	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 informations d'identification sur le cluster de stockage via l'interface de ligne de commande ONTAP ou ONTAP System Manager, 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 des pilotes SAN ONTAP avec votre installation Trident. Cette section fournit des exemples de configuration backend et des détails pour le mappage des backends à StorageClasses. "Systèmes ASA r2" diffèrent des autres systèmes ONTAP (ASA, AFF et FAS) par l'implémentation de leur couche de stockage. Ces variations ont un impact sur l'utilisation de certains paramètres comme noté. "En

savoir plus sur les différences entre les systèmes ASA r2 et les autres systèmes ONTAP". Dans la configuration du backend Trident, il n'est pas nécessaire de préciser que votre système est ASA r2. Lorsque vous sélectionnez `ontap-san` comme `storageDriverName`, Trident détecte automatiquement les systèmes ASA r2 ou autres systèmes ONTAP. Certains paramètres de configuration du backend ne s'appliquent pas aux systèmes ASA r2, comme indiqué dans le tableau ci-dessous.

REMARQUE	Seuls les `ontap-san` Le pilote (avec les protocoles iSCSI, NVMe/TCP et FC) est pris en charge pour les systèmes ASA r2.

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-san` ou `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 Exemple MetroCluster. **REMARQUE** : Si vous utilisez des identifiants « vsadmin », doit être celui du SVM ; si vous `managementLIF` utilisez des identifiants « admin », `managementLIF` doit être celui du cluster.	« 10.0.0.1 », « [2001:1234:abcd::fefe] »

Paramètre	Description	Valeur par défaut
`dataLIF`	Adresse IP de la LIF de protocole. 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]`. **Ne spécifiez pas pour iSCSI.** Trident utilise "Mappage de LUN sélectif ONTAP" pour détecter les LIFs iSCSI nécessaires pour établir une session multi-chemins. Un avertissement est généré si `dataLIF` est explicitement défini. **Omettre pour MetroCluster.** Voir la Exemple MetroCluster.	Dérivé par la SVM
`svm`	Serveur virtuel de stockage à utiliser **Omettre pour MetroCluster.** Voir Exemple MetroCluster.	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. **Non pris en charge pour FCP ou NVMe/TCP.**	`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`	« »
`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.	« »

Paramètre	Description	Valeur par défaut
`username`	Nom d'utilisateur nécessaire pour communiquer avec le cluster ONTAP . Utilisé pour l'authentification basée sur les informations d'identification. Pour l'authentification Active Directory, voir "Authentifier Trident auprès d'une SVM principale à l'aide des informations d'identification Active Directory".	« »
`password`	Mot de passe nécessaire pour communiquer avec le cluster ONTAP . Utilisé pour l'authentification basée sur les informations d'identification. Pour l'authentification Active Directory, voir "Authentifier Trident auprès d'une SVM principale à l'aide des informations d'identification Active Directory".	« »
`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. **REMARQUE** : 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. **Ne pas spécifier pour les systèmes ASA r2**.	« »

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. **Ne pas spécifier pour les systèmes ASA r2**.	« » (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`

Paramètre	Description	Valeur par défaut
`useREST`	Paramètre booléen pour utiliser les API REST ONTAP . `useREST`Lorsqu'il est réglé sur `true` Trident utilise les API REST ONTAP pour communiquer avec le backend ; lorsqu'il est défini sur `false` Trident utilise les appels ONTAPI (ZAPI) pour communiquer avec le backend. Cette fonctionnalité nécessite ONTAP 9.11.1 et versions ultérieures. De plus, le rôle de connexion ONTAP utilisé doit avoir accès au `ontapi` application. Ceci est satisfait par le prédéfini `vsadmin` et `cluster-admin` rôles. À partir de la version Trident 24.06 et ONTAP 9.15.1 ou version ultérieure, `useREST` est réglé sur `true` par défaut; changer `useREST` à `false` pour utiliser les appels ONTAPI (ZAPI). `useREST`est entièrement qualifié pour NVMe/TCP. **REMARQUE** : NVMe est pris en charge uniquement avec les API REST ONTAP et n'est pas pris en charge avec ONTAPI (ZAPI). **Si spécifié, toujours défini sur `true` pour les systèmes ASA r2**.	`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).	`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` et `ontap-san-economy` pilotes avec protocole iSCSI. De plus, pris en charge pour les systèmes ASA r2 lors de l'utilisation des protocoles iSCSI et NVMe/TCP.**	
`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 les options suivantes pour accélérer le processus de mise en forme :

- **-E nodiscard (ext3, ext4):** Ne pas tenter de supprimer des blocs au moment de mkfs (la suppression initiale des blocs est utile sur les périphériques à semi-conducteurs et le stockage clairsemé / à provisionnement fin). Cette option remplace l'option obsolète « -K » et s'applique aux systèmes de fichiers ext3 et ext4.
- **-K (xfs):** Ne tentez pas de supprimer des blocs au moment de mkfs. Cette option est applicable au système de fichiers xfs.

Authentifier Trident auprès d'une SVM principale à l'aide des informations d'identification Active Directory

Vous pouvez configurer Trident pour s'authentifier auprès d'une SVM principale à l'aide des informations d'identification Active Directory (AD). Avant qu'un compte AD puisse accéder au SVM, vous devez configurer l'accès du contrôleur de domaine AD au cluster ou au SVM. Pour l'administration du cluster avec un compte AD, vous devez créer un tunnel de domaine. Se référer à "Configurer l'accès au contrôleur de domaine Active Directory dans ONTAP" pour plus de détails.

mesures

1. Configurer les paramètres du système de noms de domaine (DNS) pour un SVM backend :

   ```
   vserver services dns create -vserver <svm_name> -dns-servers
   <dns_server_ip1>,<dns_server_ip2>
   ```

2. Exécutez la commande suivante pour créer un compte d'ordinateur pour le SVM dans Active Directory :

```
vserver active-directory create -vserver DataSVM -account-name ADSERVER1
-domain demo.netapp.com
```

3. Utilisez cette commande pour créer un utilisateur ou un groupe AD pour gérer le cluster ou le SVM

```
security login create -vserver <svm_name> -user-or-group-name
<ad_user_or_group> -application <application> -authentication-method domain
-role vsadmin
```

4. Dans le fichier de configuration du backend Trident , définissez le `username` et `password` paramètres au nom d'utilisateur ou de groupe AD et au mot de passe, respectivement.

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" **Si spécifié, défini sur `true` pour les systèmes ASA r2**.
`spaceReserve`	Mode de réservation d'espace ; « aucun » (fin) ou « volume » (épais). **Réglé sur `none` pour les systèmes ASA r2**.	« aucun »
`snapshotPolicy`	Règle Snapshot à utiliser. **Réglé sur `none` pour les systèmes ASA r2**.	« aucun »
`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 du volume réservé pour les snapshots. **Ne pas spécifier pour les systèmes ASA r2**.	« 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".	"false" **Si spécifié, définir sur `true` pour les systèmes ASA r2**.

Paramètre	Description	Valeur par défaut
`luksEncryption`	Activez le cryptage LUKS. Reportez-vous à la "Utiliser la configuration de clé unifiée Linux (LUKS)".	"" **Définir sur `false` pour les systèmes ASA r2**.
`tieringPolicy`	Politique de hiérarchisation à utiliser « aucun » **Ne pas spécifier pour les systèmes ASA r2**.	
`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'
```

REMARQUE

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 1.1 correspond aux 10 % supplémentaires ajoutés par Trident au FlexVol pour prendre en charge les métadonnées LUN . `snapshotReserve` = 5 %, et la demande PVC = 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 devrait 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.

REMARQUE	Si vous utilisez Amazon FSX on NetApp ONTAP avec Trident, NetApp vous recommande 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 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 retour en arrière transparents, préciser le SVM en utilisant `managementLIF` et omettre les `svm` paramètres. Par exemple :

```
version: 1
storageDriverName: ontap-san
managementLIF: 192.168.1.66
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>
```

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 SCSI sur FC (FCP)

Vous devez avoir un SVM configuré avec FC sur votre back-end ONTAP. Il s'agit d'une configuration back-end de base pour FC.

```
---
version: 1
backendName: fcp-backend
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_fc
username: vsadmin
password: password
sanType: fcp
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}}_{{.vo\
    lume.RequestName}}"
labels:
  cluster: ClusterA
  PVC: "{{.volume.Namespace}}_{{.volume.RequestName}}"
```

Exemple de formatoptions pour le pilote 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 les copies FlexVol volume Trident. Toutes les étiquettes présentes sur un pool virtuel sont apposées sur le volume de stockage 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 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

```yaml
---
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`

AVERTISSEMENT

- 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.
- NetApp ne recommande pas l'utilisation de l'autogrow FlexVol dans tous les pilotes ONTAP, sauf ONTAP-san. Pour contourner ce problème, Trident prend en charge l'utilisation de la réserve Snapshot et adapte les volumes FlexVol en conséquence.

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.

REMARQUE

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

Comprenez les exigences, les options d'authentification et les règles d'export pour configurer un backend ONTAP avec les pilotes ONTAP NAS. À compter de la version 25.10, NetApp Trident prend en charge "Système de stockage NetApp AFX". Les systèmes de stockage NetApp AFX diffèrent des autres systèmes ONTAP (ASA, AFF et FAS) dans l'implémentation de leur couche de stockage. Dans la configuration du backend Trident, il n'est pas nécessaire de préciser que votre système est AFX. Lorsque vous sélectionnez `ontap-nas` comme `storageDriverName`, Trident détecte automatiquement les systèmes AFX.

REMARQUE	Seuls les `ontap-nas` Le pilote (avec protocole NFS) est pris en charge pour les systèmes AFX ; le protocole SMB n'est pas pris en charge.

De formation

- Pour tous les backends ONTAP, Trident exige qu'au moins un agrégat soit attribué au 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.

AVERTISSEMENT	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

```yaml
---
version: 1
backendName: ExampleBackend
storageDriverName: ontap-nas
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
svm: svm_nfs
credentials:
  name: secret-backend-creds
```

JSON

```json
{
  "version": 1,
  "backendName": "ExampleBackend",
  "storageDriverName": "ontap-nas",
  "managementLIF": "10.0.0.1",
  "dataLIF": "10.0.0.2",
  "svm": "svm_nfs",
  "credentials": {
        "name": "secret-backend-creds"
    }
}
```

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

REMARQUE	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.

REMARQUE	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.

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
```

REMARQUE	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 volume 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.

REMARQUE	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.

AVERTISSEMENT	Vous devez configurer les protocoles NFS et SMB/CIFS au SVM pour créer un `ontap-nas-economy` volume SMB pour les clusters ONTAP sur site. La configuration de l'un de ces protocoles entraîne l'échec de la création du volume SMB.

REMARQUE	`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.

REMARQUE	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
      ```

REMARQUE	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.**	« »

Activer le SMB sécurisé

À partir de la version 25.06, NetApp Trident prend en charge le provisionnement sécurisé des volumes SMB créés à l'aide `ontap-nas` et `ontap-nas-economy` backends. Lorsque le protocole SMB sécurisé est activé, vous pouvez fournir un accès contrôlé aux partages SMB pour les utilisateurs et groupes d'utilisateurs Active Directory (AD) à l'aide de listes de contrôle d'accès (ACL).

Points à retenir

- Importation `ontap-nas-economy` les volumes ne sont pas pris en charge.
- Seuls les clones en lecture seule sont pris en charge pour `ontap-nas-economy` volumes.
- Si Secure SMB est activé, Trident ignorera le partage SMB mentionné dans le backend.
- La mise à jour de l'annotation PVC, de l'annotation de classe de stockage et du champ backend ne met pas à jour l'ACL du partage SMB.
- L'ACL de partage SMB spécifiée dans l'annotation du PVC cloné aura la priorité sur celles du PVC source.
- Assurez-vous de fournir des utilisateurs AD valides lors de l'activation du protocole SMB sécurisé. Les utilisateurs non valides ne seront pas ajoutés à la liste de contrôle d'accès.
- Si vous fournissez au même utilisateur AD dans le backend, la classe de stockage et le PVC des autorisations différentes, la priorité d'autorisation sera : PVC, classe de stockage, puis backend.
- Secure SMB est pris en charge pour `ontap-nas` importations en volume gérées et non applicable aux importations en volume non gérées.

Étapes

1. Spécifiez adAdminUser dans TridentBackendConfig comme indiqué dans l'exemple suivant :

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-nas
  managementLIF: 10.193.176.x
  svm: svm0
  useREST: true
  defaults:
    adAdminUser: tridentADtest
  credentials:
    name: backend-tbc-ontap-invest-secret
```

2. Ajoutez l'annotation dans la classe de stockage.

   Ajoutez le `trident.netapp.io/smbShareAdUser` Annotation à la classe de stockage pour activer SMB sécurisé sans faille. La valeur utilisateur spécifiée pour l'annotation `trident.netapp.io/smbShareAdUser` doit être le même que le nom d'utilisateur spécifié dans le `smbcreds` secret. Vous pouvez choisir l'une des options suivantes pour `smbShareAdUserPermission` : `full_control` , `change` , ou `read` . L'autorisation par défaut est `full_control` .

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-smb-sc
  annotations:
    trident.netapp.io/smbShareAdUserPermission: change
    trident.netapp.io/smbShareAdUser: tridentADuser
parameters:
  backendType: ontap-nas
  csi.storage.k8s.io/node-stage-secret-name: smbcreds
  csi.storage.k8s.io/node-stage-secret-namespace: trident
  trident.netapp.io/nasType: smb
provisioner: csi.trident.netapp.io
reclaimPolicy: Delete
volumeBindingMode: Immediate
```

1. Créer une PVC.

   L'exemple suivant crée un PVC :

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-pvc4
  namespace: trident
  annotations:
    trident.netapp.io/snapshotDirectory: "true"
    trident.netapp.io/smbShareAccessControl: |
      read:
        - tridentADtest
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: ontap-smb-sc
```

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 backend et des détails pour le mappage des backends à StorageClasses. À compter de la version 25.10, NetApp Trident prend en charge "Systèmes de stockage NetApp AFX". NetApp AFX les systèmes de stockage diffèrent des autres systèmes basés sur ONTAP (ASA, AFF et FAS) dans l'implémentation de leur couche de stockage.

REMARQUE	Seuls les `ontap-nas` Le pilote (avec protocole NFS) est pris en charge pour les systèmes NetApp AFX ; le protocole SMB n'est pas pris en charge.

Options de configuration du back-end

Dans la configuration du backend Trident, il n'est pas nécessaire de préciser que votre système est un NetApp AFX storage system. Lorsque vous sélectionnez `ontap-nas` comme `storageDriverName`, Trident détecte automatiquement le système de stockage AFX. Certains paramètres de configuration du backend ne sont pas applicables aux systèmes de stockage AFX.

Le tableau suivant présente les options de configuration du backend :

Paramètre	Description	Valeur par défaut
`version`		Toujours 1

Paramètre	Description	Valeur par défaut
`storageDriverName`	Nom du pilote de stockage **REMARQUE** Pour les systèmes NetApp AFX uniquement `ontap-nas` est pris en charge.	`ontap-nas`, `ontap-nas-economy` ou `ontap-nas-flexgroup`
`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 Exemple MetroCluster.	« 10.0.0.1 », « [2001:1234:abcd::fefe] »
`dataLIF`	Adresse IP de la LIF de protocole. NetApp recommande 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, ce qui vous permet de créer un DNS circulaire pour équilibrer la charge sur plusieurs dataLIFs. 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 Exemple MetroCluster.	Adresse spécifiée ou dérivée d'un SVM, si non spécifiée (non recommandé)
`svm`	Serveur virtuel de stockage à utiliser **Omettre pour MetroCluster.** Voir Exemple MetroCluster.	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"]"

Paramètre	Description	Valeur par défaut
`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 se connecter au cluster/SVM. Utilisé pour l'authentification basée sur les identifiants. Pour l'authentification Active Directory, voir "Authentifier Trident auprès d'une SVM principale à l'aide des informations d'identification Active Directory".	
`password`	Mot de passe pour se connecter au cluster/SVM. Utilisé pour l'authentification basée sur les identifiants. Pour l'authentification Active Directory, voir "Authentifier Trident auprès d'une SVM principale à l'aide des informations d'identification Active Directory".	
`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 **REMARQUE** : 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 »

Paramètre	Description	Valeur par défaut
`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. **REMARQUE** 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. **Ne pas spécifier pour les systèmes de stockage AFX.**	« »
`limitAggregateUsage`	L'approvisionnement échouera si l'utilisation dépasse ce pourcentage. **Ne s'applique pas à Amazon FSx pour ONTAP**. **Ne pas spécifier pour les systèmes de stockage AFX.**	« » (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**. **REMARQUE** : 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`	L'approvisionnement échouera si la taille du volume demandée est supérieure à cette valeur.	« » (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`	Configurer la création de volumes NFS ou SMB. Les options sont `nfs` , `smb` ou nul. La valeur nulle correspond par défaut aux volumes NFS. **Si spécifié, toujours définir sur `nfs` pour les systèmes de stockage AFX**.	`nfs`

Paramètre	Description	Valeur par défaut
`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.	« »
`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 ONTAP `.useREST`Lorsqu'il est réglé sur `true` Trident utilise les API REST ONTAP pour communiquer avec le système dorsal ; lorsqu'il est configuré pour `false` Trident utilise des appels ONTAPI (ZAPI) pour communiquer avec le backend. Cette fonctionnalité nécessite ONTAP 9.11.1 et versions ultérieures. De plus, le rôle de connexion ONTAP utilisé doit avoir accès à `ontapi` application. Ceci est satisfait par la définition prédéfinie `vsadmin` et `cluster-admin` rôles. À compter de la version Trident 24.06 et ONTAP 9.15.1 ou ultérieure, `useREST` est réglé sur `true` par défaut ; modifier `useREST` à `false` utiliser les appels ONTAPI (ZAPI). **Si spécifié, toujours définir sur `true` pour les systèmes de stockage AFX**.	`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.	

Paramètre	Description	Valeur par défaut
`adAdminUser`	Utilisateur ou groupe d'utilisateurs administrateur Active Directory avec accès complet aux partages SMB. Utilisez ce paramètre pour accorder des droits d'administrateur sur le partage SMB avec un contrôle total.	

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 »
`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 »	
`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`, `false` (Défini explicitement).
`exportPolicy`	Export policy à utiliser	« par défaut »

Paramètre	Description	Valeur 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.	« »

REMARQUE	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 désormais un nouveau calcul pour garantir que le FlexVol est correctement dimensionné avec le pourcentage snapshotReserve et le PVC. Lorsqu'un utilisateur demande un PVC, Trident crée le FlexVol d'origine avec plus d'espace grâce à ce nouveau calcul.

Ce calcul garantit que l'utilisateur reçoit l'espace inscriptible qu'il a demandé sur le PVC, et non un espace inférieur à celui demandé. Avant la version 21.07, lorsqu'un utilisateur demandait un PVC (par exemple, 5 Gio), avec un snapshotReserve à 50 %, il ne recevait que 2,5 Gio d'espace inscriptible. En effet, l'utilisateur a demandé le volume entier et `snapshotReserve` est un pourcentage de cela. Avec Trident 21.07, ce que l'utilisateur demande, c'est l'espace inscriptible, et Trident définit cet espace. `snapshotReserve` nombre en pourcentage du volume total. Cela ne s'applique pas à `ontap-nas-economy` . Consultez l'exemple suivant pour voir comment cela fonctionne :

Le calcul est le suivant :

```
Total volume size = <PVC requested size> / (1 - (<snapshotReserve
percentage> / 100))
```

Pour snapshotReserve = 50 % et une demande PVC = 5 Gio, la taille totale du volume est de 5/0,5 = 10 Gio et la taille disponible est de 5 Gio, ce qui correspond à ce que l'utilisateur a demandé dans la demande PVC . `volume show` la commande devrait afficher des résultats similaires à cet exemple :

Les backends existants des installations précédentes provisionneront les volumes comme expliqué ci-dessus lors de la mise à niveau de Trident. Pour les volumes créés avant la mise à niveau, vous devez les redimensionner afin que la modification soit prise en compte. Par exemple, un PVC de 2 Gio avec `snapshotReserve=50` Auparavant, le volume fournissait 1 Gio d'espace inscriptible. Par exemple, le redimensionnement à 3 Gio permet à l'application de disposer de 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.

REMARQUE	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}}_{{.vo\
    lume.RequestName}}"
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"
```

- Le `protection-silver-creditpoints-20k` StorageClass sera associé au troisième pool virtuel dans le `ontap-nas-flexgroup` backend. Il s'agit du seul pool 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 dataLIF après la configuration initiale en exécutant la commande suivante pour fournir le nouveau fichier JSON back-end avec une dataLIF mise à jour.

```
tridentctl update backend <backend-name> -f <path-to-backend-json-file-
with-updated-dataLIF>
```

REMARQUE	Si des ESV sont associées à un ou plusieurs pods, vous devez arrêter tous les pods correspondants, puis les remonter pour que la nouvelle dataLIF prenne effet.

Exemples de PME sécurisées

Configuration du backend avec le pilote ontap-nas

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-nas
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-nas
  managementLIF: 10.0.0.1
  svm: svm2
  nasType: smb
  defaults:
    adAdminUser: tridentADtest
  credentials:
    name: backend-tbc-ontap-invest-secret
```

Configuration du backend avec le pilote ontap-nas-economy

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-nas
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-nas-economy
  managementLIF: 10.0.0.1
  svm: svm2
  nasType: smb
  defaults:
    adAdminUser: tridentADtest
  credentials:
    name: backend-tbc-ontap-invest-secret
```

Configuration du backend avec pool de stockage

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-nas
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-nas
  managementLIF: 10.0.0.1
  svm: svm0
  useREST: false
  storage:
  - labels:
      app: msoffice
    defaults:
      adAdminUser: tridentADuser
  nasType: smb
  credentials:
    name: backend-tbc-ontap-invest-secret
```

Exemple de classe de stockage avec le pilote ontap-nas

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-smb-sc
  annotations:
    trident.netapp.io/smbShareAdUserPermission: change
    trident.netapp.io/smbShareAdUser: tridentADtest
parameters:
  backendType: ontap-nas
  csi.storage.k8s.io/node-stage-secret-name: smbcreds
  csi.storage.k8s.io/node-stage-secret-namespace: trident
  trident.netapp.io/nasType: smb
provisioner: csi.trident.netapp.io
reclaimPolicy: Delete
volumeBindingMode: Immediate
```

REMARQUE	Assurez-vous d'ajouter `annotations` pour activer le protocole SMB sécurisé. Le protocole SMB sécurisé ne fonctionne pas sans les annotations, quelles que soient les configurations définies dans le backend ou le PVC.

Exemple de classe de stockage avec le pilote ontap-nas-economy

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-smb-sc
  annotations:
    trident.netapp.io/smbShareAdUserPermission: change
    trident.netapp.io/smbShareAdUser: tridentADuser3
parameters:
  backendType: ontap-nas-economy
  csi.storage.k8s.io/node-stage-secret-name: smbcreds
  csi.storage.k8s.io/node-stage-secret-namespace: trident
  trident.netapp.io/nasType: smb
provisioner: csi.trident.netapp.io
reclaimPolicy: Delete
volumeBindingMode: Immediate
```

Exemple de PVC avec un seul utilisateur AD

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-pvc4
  namespace: trident
  annotations:
    trident.netapp.io/smbShareAccessControl: |
      change:
        - tridentADtest
      read:
        - tridentADuser
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: ontap-smb-sc
```

Exemple de PVC avec plusieurs utilisateurs AD

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-test-pvc
  annotations:
    trident.netapp.io/smbShareAccessControl: |
      full_control:
        - tridentTestuser
        - tridentuser
        - tridentTestuser1
        - tridentuser1
      change:
        - tridentADuser
        - tridentADuser1
        - tridentADuser4
        - tridentTestuser2
      read:
        - tridentTestuser2
        - tridentTestuser3
        - tridentADuser2
        - tridentADuser3
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
```

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 exécute des systèmes de fichiers alimentés par le système d'exploitation de stockage NetApp ONTAP. Il fournit les fonctionnalités, les performances et l'administration d'ONTAP avec l'évolutivité et la simplicité opérationnelle d'AWS. Un système de fichiers est la ressource principale dans Amazon FSx et est analogue à un cluster ONTAP sur site. Chaque système de fichiers contient une ou plusieurs machines virtuelles de stockage (SVM), et chaque SVM contient un ou plusieurs volumes qui stockent des fichiers et des répertoires. Cette intégration permet aux clusters Kubernetes exécutés dans Amazon Elastic Kubernetes Service (EKS) de provisionner des volumes persistants basés sur ONTAP pour les charges de travail bloc et fichier.

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".

REMARQUE	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 uniquement en charge les volumes SMB montés sur des pods exécutés sur des nœuds Windows. Consultez "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.

REMARQUE	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 .

Utilisation simultanée des pilotes Trident SAN/iSCSI et EBS-CSI

Si vous prévoyez d'utiliser des pilotes ontap-san (par exemple, iSCSI) avec AWS (EKS, ROSA, EC2 ou toute autre instance), la configuration multi-chemin requise sur les nœuds peut entrer en conflit avec le pilote CSI Amazon Elastic Block Store (EBS). Pour garantir que le multivoie fonctionne sans interférer avec les disques EBS sur le même nœud, vous devez exclure EBS dans votre configuration de multivoie. Cet exemple montre un `multipath.conf` fichier qui inclut les paramètres Trident requis tout en excluant les disques EBS du multi-accès :

```
defaults {
    find_multipaths no
}
blacklist {
    device {
        vendor "NVME"
        product "Amazon Elastic Block Store"
    }
}
```

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.

AVERTISSEMENT	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"

Ami (Amazon machine Images) testé

Le cluster EKS prend en charge plusieurs systèmes d'exploitation, mais AWS a optimisé certains ami (Amazon machine image) pour les conteneurs et EKS. Les AMI suivants ont été testés avec NetApp Trident 25.02.

AMI	NAS	Économie NAS	ISCSI	économie iSCSI
AL2023_x86_64_STANDARD	Oui.	Oui.	Oui.	Oui.
AL2_x86_64	Oui.	Oui.	Oui*	Oui*
BOTTLEROCKET_x86_64	Oui**	Oui.	S/O	S/O
AL2023_ARM_64_STANDARD	Oui.	Oui.	Oui.	Oui.
AL2_ARM_64	Oui.	Oui.	Oui*	Oui*

BOTTLEROCKET_ARM_64	Oui**	Oui.	S/O	S/O

- * Impossible de supprimer le PV sans redémarrer le nœud
- ** Ne fonctionne pas avec NFSv3 avec Trident version 25.02.

REMARQUE	Si votre ami souhaité n'est pas répertorié ici, cela ne signifie pas qu'il n'est pas pris en charge ; cela signifie simplement qu'il n'a pas été testé. Cette liste sert de guide pour les AMI dont on sait qu'ils fonctionnent.

Tests effectués avec :

- Version EKS : 1.32
- Méthode d'installation : Helm 25.06 et en tant que module complémentaire AWS 25.06
- Pour NAS, NFS v3 et NFS v4.1 ont été testés.
- Pour le SAN, iSCSI uniquement a été testé, pas NVMe-of.

Tests effectués :

- Créer : classe de stockage, pvc, pod
- Suppression : pod, pvc (normal, qtree/lun – économique, NAS avec sauvegarde AWS)

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.

REMARQUE	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 secrets Manager

Comme Trident émettra des API pour un vServer FSX afin de gérer le stockage pour vous, il aura besoin d'informations d'identification pour le faire. La façon sécurisée de transmettre ces informations d'identification est de passer par un secret AWS secrets Manager. Par conséquent, si vous n'en avez pas déjà un, vous devrez créer un secret AWS secrets Manager contenant les informations d'identification du compte vsadmin.

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

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

Créer une politique IAM

Trident a également besoin des autorisations AWS pour s'exécuter correctement. Par conséquent, vous devez créer une stratégie qui donne à Trident les autorisations dont elle a besoin.

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
Secrets manager"
```

Exemple JSON de règles :

```
{
  "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 une identité de pod ou un rôle IAM pour l'association de comptes de service (IRSA)

Vous pouvez configurer un compte de service Kubernetes pour qu'il assume un rôle AWS Identity and Access Management (IAM) avec l'identité de pod EKS ou le rôle IAM pour l'association de comptes de service (IRSA). Tous les pods configurés pour utiliser ce compte de service peuvent alors accéder à tous les services AWS auxquels ce rôle est autorisé.

Identité du pod

Les associations d'identité de pod Amazon EKS offrent la possibilité de gérer les informations d'identification de vos applications, de la même manière que les profils d'instance Amazon EC2 fournissent des informations d'identification aux instances Amazon EC2.

Installez Pod Identity sur votre cluster EKS :

Vous pouvez créer une identité de pod via la console AWS ou à l'aide de la commande AWS CLI suivante :

```
aws eks create-addon --cluster-name <EKS_CLUSTER_NAME> --addon-name
eks-pod-identity-agent
```

Pour plus d'informations, reportez-vous à "Configurer l'agent d'identité du pod Amazon EKS" .

Créer trust-relationship.json:

Créez le fichier trust-relationship.json pour permettre au principal du service EKS d'assumer ce rôle pour l'identité du pod. Créez ensuite un rôle avec la politique de confiance suivante :

```
aws iam create-role \
  --role-name fsxn-csi-role --assume-role-policy-document file://trust-
relationship.json \
  --description "fsxn csi pod identity role"
```

fichier trust-relationship.json:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "pods.eks.amazonaws.com"
      },
      "Action": [
        "sts:AssumeRole",
        "sts:TagSession"
      ]
    }
  ]
}
```

Attachez la politique de rôle au rôle IAM:

Attachez la politique de rôle de l'étape précédente au rôle IAM qui a été créé :

```
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:111122223333:policy/fsxn-csi-policy \
  --role-name fsxn-csi-role
```

Créer une association d'identité de pod:

Créer une association d'identité de pod entre le rôle IAM et le compte de service Trident (trident-controller)

```
aws eks create-pod-identity-association \
  --cluster-name <EKS_CLUSTER_NAME> \
  --role-arn arn:aws:iam::111122223333:role/fsxn-csi-role \
  --namespace trident --service-account trident-controller
```

Rôle IAM pour l'association de comptes de service (IRSA)

Utilisation de l'AWS CLI :

```
aws iam create-role --role-name AmazonEKS_FSxN_CSI_DriverRole \
  --assume-role-policy-document file://trust-relationship.json
```

fichier trust-relationship.json :

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws:iam::<account_id>:oidc-
provider/<oidc_provider>"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringEquals": {
          "<oidc_provider>:aud": "sts.amazonaws.com",
          "<oidc_provider>:sub":
"system:serviceaccount:trident:trident-controller"
        }
      }
    }
  ]
}
```

Mettez à jour les valeurs suivantes dans le `trust-relationship.json` fichier :

- **<account_id>** - votre ID de compte AWS
- **<oidc_provider>** - l'OIDC de votre cluster EKS. Vous pouvez obtenir le fournisseur oidc_Provider en exécutant :

```
aws eks describe-cluster --name my-cluster --query
"cluster.identity.oidc.issuer"\
  --output text | sed -e "s/^https:\/\///"
```

Joindre le rôle IAM à la politique IAM :

Une fois le rôle créé, reliez la stratégie (créée à l'étape ci-dessus) au rôle à l'aide de la commande suivante :

```
aws iam attach-role-policy --role-name my-role --policy-arn <IAM policy
ARN>
```

Vérifier que le fournisseur OICD est associé :

Vérifiez que votre fournisseur OIDC est associé à votre cluster. Vous pouvez le vérifier à l'aide de la commande suivante :

```
aws iam list-open-id-connect-providers | grep $oidc_id | cut -d "/" -f4
```

Si la sortie est vide, utiliser la commande suivante pour associer IAM OIDC à votre cluster :

```
eksctl utils associate-iam-oidc-provider --cluster $cluster_name
--approve
```

Si vous utilisez eksctl, utilisez l'exemple suivant pour créer 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 <IAM-Policy ARN> --approve
```

Installation de Trident

Trident simplifie la gestion du stockage Amazon FSx for NetApp ONTAP dans Kubernetes afin de permettre à vos développeurs et administrateurs de se concentrer sur le déploiement des applications. Vous pouvez installer Trident en utilisant l'une des méthodes suivantes :

- Gouvernail
- Module complémentaire EKS

Si vous souhaitez utiliser la fonctionnalité snapshot, installez le module complémentaire CSI snapshot Controller. Pour plus d'informations, reportez-vous à la section "Activer la fonctionnalité snapshot pour les volumes CSI" .

Installez Trident via Helm

Identité du pod

1. Ajout du référentiel Trident Helm :

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

2. Installez Trident en utilisant l'exemple suivant :

```
helm install trident-operator netapp-trident/trident-operator --version 100.2502.1 --namespace trident --create-namespace
```

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-
100.2502.0    25.02.0
```

Association de comptes de service (IRSA)

1. Ajout du référentiel Trident Helm :

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

2. Définissez les valeurs pour **fournisseur de cloud** et **identité cloud** :

```
helm install trident-operator netapp-trident/trident-operator
--version 100.2502.1  \
--set cloudProvider="AWS" \
--set cloudIdentity="'eks.amazonaws.com/role-arn:
arn:aws:iam::<accountID>:role/<AmazonEKS_FSxN_CSI_DriverRole>'" \
--namespace trident \
--create-namespace
```

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-
100.2510.0    25.10.0
```

REMARQUE

Si vous prévoyez d'utiliser iSCSI, assurez-vous qu'il est activé sur votre machine cliente. Si vous utilisez le système d'exploitation du nœud Worker AL2023, vous pouvez automatiser l'installation du client iSCSI en ajoutant le paramètre node prep à l'installation de Helm :

```
helm install trident-operator netapp-trident/trident-
operator --version 100.2502.1 --namespace trident --create
-namespace --set nodePrep={iscsi}
```

Installez Trident via le module complémentaire EKS

Le module complémentaire Trident EKS inclut les derniers correctifs de sécurité et de bogues, et est validé par AWS pour une utilisation 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 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 Trident pour AWS

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, sélectionnez **clusters**.
3. Sélectionnez le nom du cluster pour lequel vous souhaitez configurer le module complémentaire NetApp Trident CSI.
4. Sélectionnez **Compléments**, puis **obtenir plus de modules complémentaires**.
5. Suivez ces étapes pour sélectionner le module complémentaire :
   a. Faites défiler jusqu'à la section **Modules complémentaires AWS Marketplace** et saisissez **« Trident »** dans la zone de recherche.
   b. Cochez la case dans le coin supérieur droit de la boîte Trident by NetApp.
   c. Sélectionnez **Suivant**.
6. Sur la page **configurer les compléments sélectionnés**, procédez comme suit :

   REMARQUE | **Ignorez ces étapes si vous utilisez l'association d'identité de pod.**

   a. Sélectionnez la **version** que vous souhaitez utiliser.
   b. Si vous utilisez l'authentification IRSA, assurez-vous de définir les valeurs de configuration disponibles dans les paramètres de configuration facultatifs :
      - Sélectionnez la **version** que vous souhaitez utiliser.
      - Suivez le **schéma de configuration du module complémentaire** et définissez le paramètre **configurationValues** dans la section **Valeurs de configuration** sur le rôle-arn que vous avez créé à l'étape précédente (la valeur doit être au format suivant) :

```
{

  "cloudIdentity": "'eks.amazonaws.com/role-arn: <role ARN>'",
  "cloudProvider": "AWS"

}
```

   +
   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.
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. Créer le `add-on.json` déposer:

Pour l'identité du pod, utilisez le format suivant :

REMARQUE Utilisez le

```
{
  "clusterName": "<eks-cluster>",
  "addonName": "netapp_trident-operator",
  "addonVersion": "v25.6.0-eksbuild.1",
}
```

Pour l'authentification IRSA, utilisez le format suivant :

```
{
  "clusterName": "<eks-cluster>",
  "addonName": "netapp_trident-operator",
  "addonVersion": "v25.6.0-eksbuild.1",
  "serviceAccountRoleArn": "<role ARN>",
  "configurationValues": {
    "cloudIdentity": "'eks.amazonaws.com/role-arn: <role ARN>'",
    "cloudProvider": "AWS"
  }
}
```

REMARQUE Remplacer `<role ARN>` par l'ARN du rôle créé à l'étape précédente.

2. Installez le module complémentaire Trident EKS.

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

eksctl

L'exemple de commande suivant installe le module complémentaire Trident EKS :

```
eksctl create addon --name netapp_trident-operator --cluster
<cluster_name> --force
```

Mettez à jour le module complémentaire Trident EKS

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, sélectionnez **clusters**.
3. Sélectionnez le nom du cluster pour lequel vous souhaitez mettre à jour le module complémentaire NetApp Trident CSI.
4. Sélectionnez l'onglet **Compléments**.
5. Sélectionnez **Trident by NetApp**, puis **Edit**.
6. Sur la page **configurer Trident par NetApp**, procédez comme suit :
   a. Sélectionnez la **version** que vous souhaitez utiliser.
   b. Développez les **Paramètres de configuration facultatifs** et modifiez-les si nécessaire.
   c. Sélectionnez **Enregistrer les modifications**.

CLI AWS

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

```
aws eks update-addon --cluster-name <eks_cluster_name> --addon-name
netapp_trident-operator --addon-version v25.6.0-eksbuild.1 \
  --service-account-role-arn <role-ARN> --resolve-conflict preserve \
  --configuration-values "{\"cloudIdentity\":
\"'eks.amazonaws.com/role-arn: <role ARN>'\"}"
```

eksctl

- 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    v25.6.0-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
v25.6.0-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".

Désinstallez/supprimez le module complémentaire 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** – NetApp vous recommande 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.

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

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, sélectionnez **clusters**.
3. Sélectionnez le nom du cluster pour lequel vous souhaitez supprimer le module complémentaire NetApp Trident CSI.
4. Sélectionnez l'onglet **Compléments**, puis sélectionnez **Trident by NetApp**.*
5. Sélectionnez **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. Sélectionnez **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
```

eksctl

La commande suivante désinstalle le module complémentaire Trident EKS :

```
eksctl delete addon --cluster K8s-arm --name netapp_trident-operator
```

Configurer une classe de stockage

La "Objet classe de stockage Kubernetes" identifie un provisionneur et lui indique comment provisionner les volumes. Cette section montre comment configurer un objet Kubernetes StorageClass qui spécifie Trident comme provisionneur.

Créer un objet StorageClass

Lorsque vous créez un StorageClass pour FSx for ONTAP, Trident créera automatiquement la configuration backend.

REMARQUE	Si vous souhaitez configurer manuellement le stockage backend, veuillez vous référer à la section [create-a-kubernetes-storageclass-without-automatic-backend-configuration] pour savoir comment créer séparément le backend Trident et la classe de stockage.

Spécifiez les paramètres requis StorageClass

Les trois paramètres suivants doivent être définis lors de la création d'un StorageClass :

Paramètre	Obligatoire	Type	Description
`fsxFilesystemID`	Oui.	chaîne	ID du système de fichiers FSx for NetApp ONTAP
`storageDriverName`	Oui.	chaîne	Pilote de stockage Trident (par exemple, `ontap-nas` ou `ontap-san`)
`credentialsName`	Oui.	chaîne	Nom du secret Kubernetes qui contient les informations d'identification FSx for ONTAP

Spécifiez les paramètres optionnels

Vous pouvez transmettre des paramètres backend optionnels via le StorageClass. Définissez toutes les valeurs optionnelles sous forme de chaînes dans la section StorageClass `parameters`. Pour une liste complète des paramètres du backend, voir : "Configuration du backend FSx for NetApp ONTAP".

Exemples de fichiers de configuration StorageClass.

L'exemple suivant montre un StorageClass qui déclenche une configuration automatique du backend.

YAML

```yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-fsx-demo
  annotations:
    description: "Demo StorageClass for FSx for NetApp ONTAP"
provisioner: csi.trident.netapp.io
parameters:
  fsxFilesystemID: "fs-0abc123"
  storageDriverName: "ontap-nas"
  credentialsName: trident-fsx-credentials
allowVolumeExpansion: true
reclaimPolicy: Delete
volumeBindingMode: Immediate
```

JSON

```json
{
  "apiVersion": "storage.k8s.io/v1",
  "kind": "StorageClass",
  "metadata": {
    "name": "ontap-fsx-demo",
    "annotations": {
      "description": "Demo StorageClass for FSx for NetApp ONTAP"
    }
  },
  "provisioner": "csi.trident.netapp.io",
  "parameters": {
    "fsxFilesystemID": "fs-0abc123",
    "storageDriverName": "ontap-nas",
    "credentialsName": "trident-fsx-credentials"
  },
  "allowVolumeExpansion": true,
  "reclaimPolicy": "Delete",
  "volumeBindingMode": "Immediate"
}
```

Créez le StorageClass

Une fois que vous avez créé votre fichier de configuration, exécutez la commande suivante pour créer la classe de stockage.

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

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
```

Après avoir appliqué le StorageClass, Trident crée automatiquement le backend. Vous pouvez ensuite créer des PersistentVolumeClaims qui référencent ce StorageClass.

Vérifier l'état de la configuration du backend

Trident enregistre le résultat de la création du backend dans les annotations StorageClass.

Annotation	Description
`trident.netapp.io/configuratorStatus`	Résultat de la configuration (`Success` ou `Failure`)
`trident.netapp.io/configuratorMessage`	Message d'état ou d'erreur détaillé
`trident.netapp.io/configuratorName`	Nom de la ressource du configurateur interne
`trident.netapp.io/managed`	Indique que le StorageClass est géré par Trident
`trident.netapp.io/additionalStoragePools`	Pools de stockage créés pour ce backend

Pour vérifier l'état, exécutez :

```
kubectl get storageclass ontap-fsx-demo -o yaml
```

Confirmez que `trident.netapp.io/configuratorStatus` est défini sur `Success`. Si la valeur est `Failure`, examinez `trident.netapp.io/configuratorMessage` pour l'erreur.

Ajouter des systèmes de fichiers FSxN supplémentaires

Si vous avez besoin de stockage supplémentaire tout en continuant à utiliser le même StorageClass, ajoutez des identifiants de système de fichiers FSxN supplémentaires.

Modifiez le StorageClass et ajoutez l'annotation suivante :

```
metadata:
  annotations:
    trident.netapp.io/additionalFsxNFileSystemID: '["fs-
xxxxxxxxxxxxxxxxxx"]'
```

Une fois la modification appliquée, Trident met à jour la configuration du backend et met à jour les annotations StorageClass.

Considérations et limitations opérationnelles

- La suppression d'un StorageClass qui utilise la configuration automatique du backend supprime généralement le backend Trident associé. Cela peut perturber la connectivité du stockage et interrompre les charges de travail en cours. Validez l'impact avant de supprimer un StorageClass géré.
- La configuration automatique du backend est prise en charge uniquement pour AWS FSx pour NetApp ONTAP.

Créer un StorageClass Kubernetes sans configuration automatique du backend

Si vous souhaitez créer le backend Trident et le StorageClass séparément, suivez ces étapes.

Comprendre le fonctionnement de la configuration automatique du backend

Trident déduit la configuration du backend à partir de la définition de la StorageClass. Lorsque vous appliquez la StorageClass, Trident valide les paramètres requis, crée le backend et annote la StorageClass avec le statut.

Trident crée le VolumeSnapshotClass une seule fois. Trident réutilise le même VolumeSnapshotClass pour les StorageClasses suivants.

Créer le backend Trident

Pour créer un backend Trident, vous devez créer un fichier de configuration au format JSON ou YAML. Le fichier doit spécifier le type de stockage souhaité (NAS ou SAN), le système de fichiers, la SVM à utiliser et la méthode d'authentification. L'exemple suivant montre comment définir un stockage basé sur NAS et utiliser un secret AWS pour stocker les informations d'identification de la SVM que vous souhaitez utiliser :

YAML

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-nas
  namespace: trident
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"
    "namespace": "trident"
  },
  "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"
    }
  }
}
```

Détails du pilote FSX pour ONTAP

Vous pouvez intégrer Trident avec Amazon FSX for NetApp ONTAP à l'aide des pilotes suivants :

Nom du pilote	Description
`ontap-san`	Chaque PV provisionné est un LUN au sein de son propre volume Amazon FSx for NetApp ONTAP. Recommandé pour le stockage bloc.
`ontap-nas`	Chaque PV provisionné est un volume Amazon FSx for NetApp ONTAP complet. Recommandé pour NFS et SMB.
`ontap-san-economy`	Chaque PV provisionné est un LUN avec un nombre configurable de LUN par Amazon FSx for NetApp ONTAP volume.
`ontap-nas-economy`	Chaque PV provisionné est un qtree, avec un nombre configurable de qtrees par volume Amazon FSx for NetApp ONTAP.
`ontap-nas-flexgroup`	Chaque PV provisionné est un volume Amazon FSx for NetApp ONTAP FlexGroup complet.

Pour plus d'informations sur le pilote, reportez-vous à la section "Pilotes NAS" et "Pilotes SAN".

Créer le backend

Après avoir créé le fichier de configuration, exécutez les commandes suivantes pour créer et valider la configuration du backend Trident (TBC) :

- Créez la configuration Trident backend (TBC) à partir du fichier yaml et exécutez la commande suivante :

```
kubectl create -f backendconfig.yaml -n trident
```

```
tridentbackendconfig.trident.netapp.io/backend-tbc-ontap-nas created
```

- Vérifiez que la configuration du back-end Trident (TBC) a été créée avec succès :

```
Kubectl get tbc -n trident
```

```
NAME                          BACKEND NAME          BACKEND UUID
PHASE   STATUS

backend-tbc-ontap-nas         tbc-ontap-nas         933e0071-66ce-4324-
b9ff-f96d916ac5e9   Bound   Success
```

Pour plus d'informations sur les autres options de configuration, consultez la [Backend-advanced-configuration-and-examples] section ci-dessous.

Configurer une classe de stockage sans configuration automatique du backend

Voici des exemples de configurations de classes de stockage à utiliser avec Trident et FSx pour ONTAP.

Classe de stockage pour NFS

Vous pouvez utiliser cet exemple pour configurer StorageClass pour des volumes utilisant NFS (reportez-vous à la section Attributs Trident ci-dessous pour la liste complète des attributs) :

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

Classe de stockage pour iSCSI

Utilisez cet exemple pour configurer StorageClass pour des volumes utilisant iSCSI :

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

Classe de stockage utilisant NFSv3 et AWS Bottlerocket

Pour provisionner des volumes NFSv3 sur AWS BottlerRocket, ajoutez les éléments requis `mountOptions` à la classe de stockage :

```
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"
mountOptions:
  - nfsvers=3
  - nolock
```

Attributs de Trident StorageClass

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, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, solidfire-san, 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

Attribut	Type	Valeurs	Offre	Demande	Pris en charge par
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
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 ONTAP Select ou les systèmes FSx for ONTAP

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 la classe de stockage

Une fois que vous avez configuré le StorageClass, vous pouvez le créer dans Kubernetes.

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
```

Volumes de provisionnement pour PME

Vous pouvez provisionner des volumes SMB à l'aide du `ontap-nas` driver. Cependant, pour ce faire, vous devez effectuer ces étapes : "Préparez-vous au provisionnement des volumes SMB".

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
`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] »

Paramètre	Description	Exemple
`dataLIF`	Adresse IP de la LIF de protocole. **ONTAP NAS drivers** : NetApp recommande de spécifier dataLIF. Si elle n'est pas fournie, Trident récupère les dataLIF depuis la SVM. Vous pouvez spécifier un nom de domaine pleinement qualifié (FQDN) à utiliser pour les opérations de montage NFS, ce qui vous permet de créer un DNS à répartition de charge (round-robin) entre plusieurs dataLIF. Peut être modifié après le paramétrage initial. **ONTAP SAN drivers** : ne pas spécifier pour iSCSI. Trident utilise ONTAP Selective LUN Map pour découvrir les LIF iSCSI nécessaires à l'établissement d'une session multi-chemin. Un avertissement est généré si dataLIF est explicitement défini. Peut être défini pour utiliser des adresses IPv6 si Trident a été installé avec le paramètre IPv6. Les adresses IPv6 doivent être définies entre crochets, par exemple [28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555].	
`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	« »

Paramètre	Description	Exemple
`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`
`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. Limite également la taille maximale des volumes gérés pour les qtrees et les LUN, et `qtreesPerFlexvol` permet de personnaliser le nombre maximal de qtrees par FlexVol volume	« » (non appliqué par défaut)
`lunsPerFlexvol`	Le nombre maximal de LUN par FlexVol volume doit être compris dans la plage [50, 200]. SAN uniquement.	« 100 »

Paramètre	Description	Exemple
`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
`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.	« »
`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 qtree par FlexVol volume, 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`

Paramètre	Description	Exemple
`useREST`	Paramètre booléen pour utiliser les API REST de ONTAP. 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.	"" "" ""
`credentials`	Spécifiez les informations d'identification de SVM FSX à stocker dans AWS secrets Manager. - `name`: Amazon Resource Name (ARN) du secret, qui contient les références de SVM. - `type`: Réglé sur `awsarn`. Pour plus d'informations, reportez-vous à la section "Créez un secret AWS secrets Manager" .	

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 ; « aucun » (fin) ou « volume » (épais)	`none`
`snapshotPolicy`	Règle Snapshot à utiliser	`none`

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 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 snapshots « 0 »	Si `snapshotPolicy` est `none`, `else` ""
`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 chiffrement LUKS. Consultez "Utiliser la configuration de clé unifiée Linux (LUKS)". SAN uniquement.	« »
`tieringPolicy`	Règle de hiérarchisation à utiliser `none`	
`unixPermissions`	Mode pour les nouveaux volumes. **Laisser vide pour les volumes SMB.**	« »

Paramètre	Description	Valeur 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`.

Configurer un PVC

Cette section explique comment créer un PersistentVolumeClaim (PVC) qui utilise la StorageClass Kubernetes configurée pour demander un PV. En cas de succès, vous pourrez ensuite monter le PV sur un pod.

Créer la PVC

Un "*PersistentVolumeClaim*" (PVC) est une demande d'accès au PersistentVolume sur le cluster. Le PVC peut être configuré pour demander un espace de stockage d'une certaine taille ou un certain mode d'accès. En utilisant la StorageClass associée, l'administrateur du cluster peut contrôler plus que la taille et le mode d'accès du PersistentVolume—comme les performances ou le niveau de service.

Après avoir créé le backend Trident et le StorageClass, vous pouvez créer un PVC. Après la création du PVC, vous pouvez monter le volume dans un pod.

Exemples de manifestes

Les exemples suivants illustrent les options de configuration de base du PVC.

PVC avec accès RWX

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: ontap-gold
```

Exemple de PVC utilisant iSCSI

Cet exemple montre un PVC de base pour iSCSI avec accès RWO associé à une StorageClass nommée `protection-gold`.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc-san
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: protection-gold
```

Créer PVC

Étapes

1. Créer la PVC.

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

2. 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.

Déployer une application

Une fois la classe de stockage et la demande de volume persistant créées, vous pouvez monter le volume persistant sur un pod. Cette section répertorie l'exemple de commande et de configuration permettant d'attacher le volume persistant à un pod.

Déployer une 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
```

REMARQUE 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 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
```

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

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

Configurer le module complémentaire Trident EKS sur un cluster EKS

NetApp Trident rationalise la gestion du stockage Amazon FSX for NetApp ONTAP dans Kubernetes pour que vos développeurs et administrateurs puissent donner la priorité au déploiement d'applications. Le module complémentaire NetApp Trident EKS inclut les

derniers correctifs de sécurité et de bogues, et est validé par AWS pour une utilisation 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.

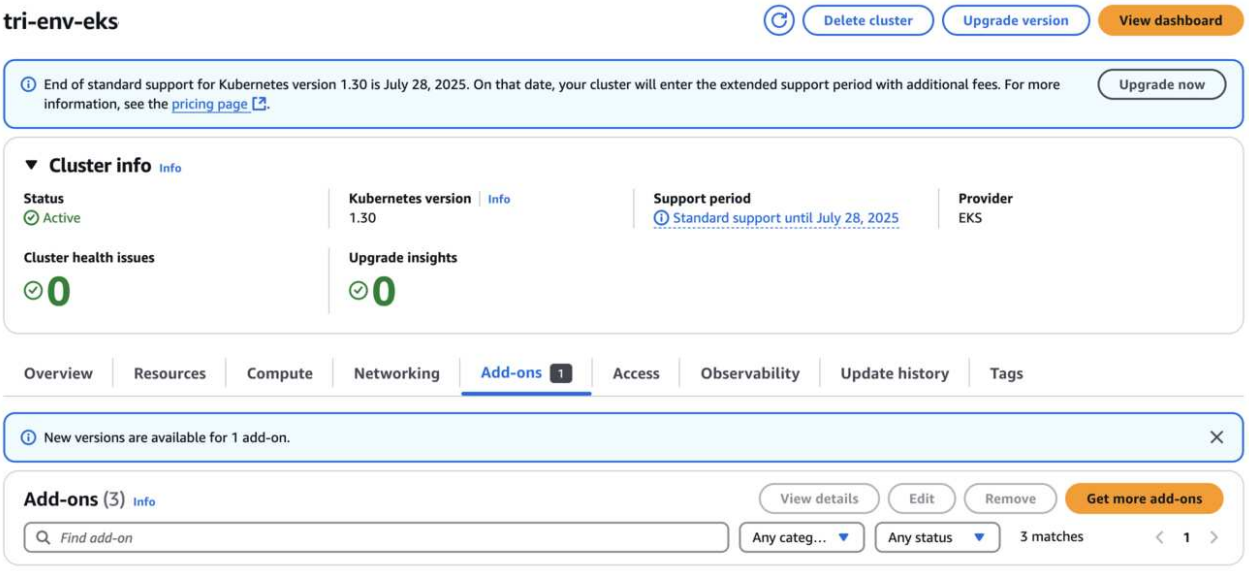
Prérequis

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

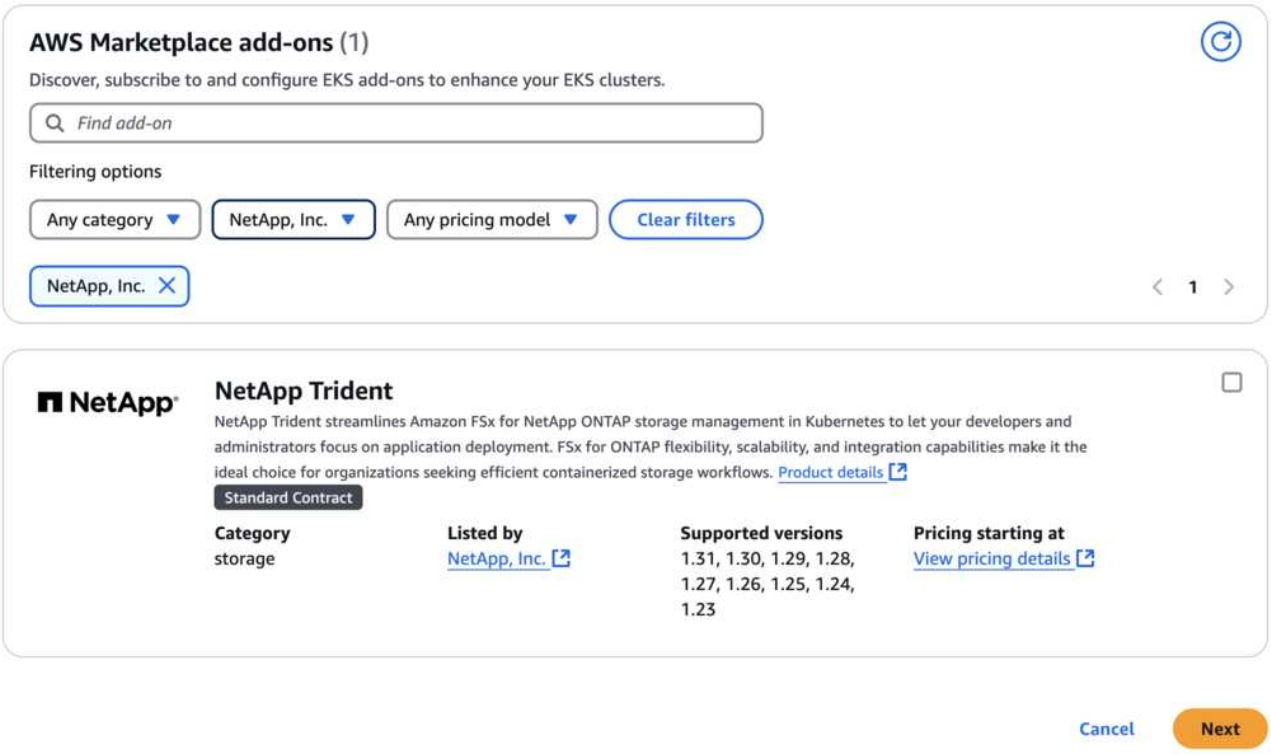
- Un compte de cluster Amazon EKS avec des autorisations d'utilisation de modules complémentaires. Reportez-vous à la "Add-ons Amazon EKS".
- Autorisations AWS sur AWS Marketplace :
  `"aws-marketplace:ViewSubscriptions",`
  `"aws-marketplace:Subscribe",`
  `"aws-marketplace:Unsubscribe`
- Type ami : 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

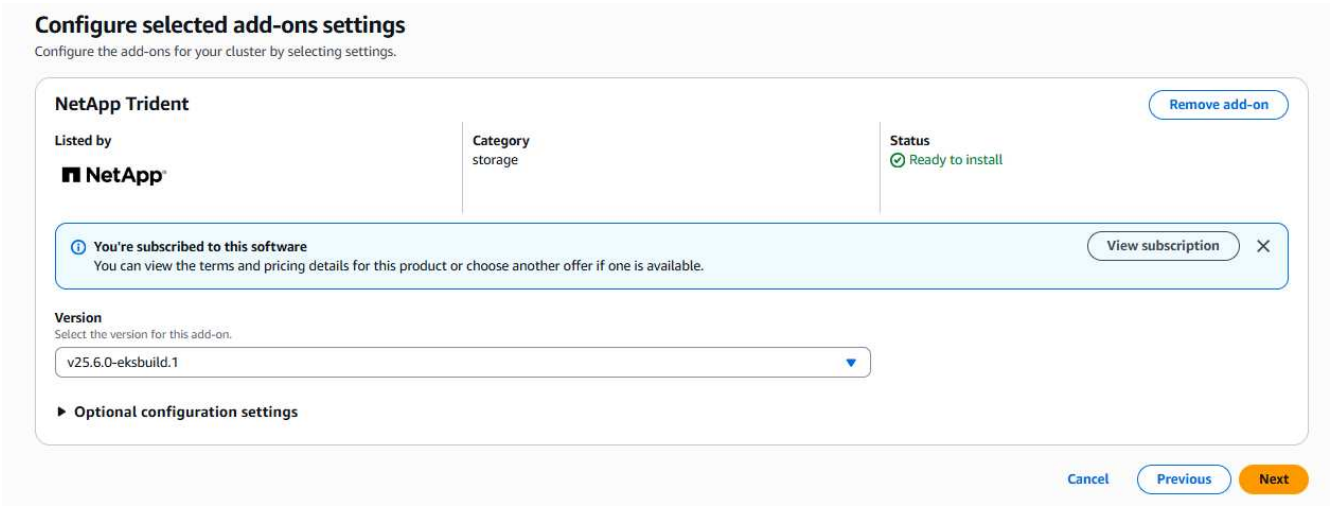
1. Veillez à créer un rôle IAM et un code AWS secret pour permettre aux pods d'EKS d'accéder aux ressources AWS. Pour obtenir des instructions, reportez-vous à la section "Créez un rôle IAM et un code secret AWS".
2. Sur votre cluster EKS Kubernetes, accédez à l'onglet **Add-ons**.



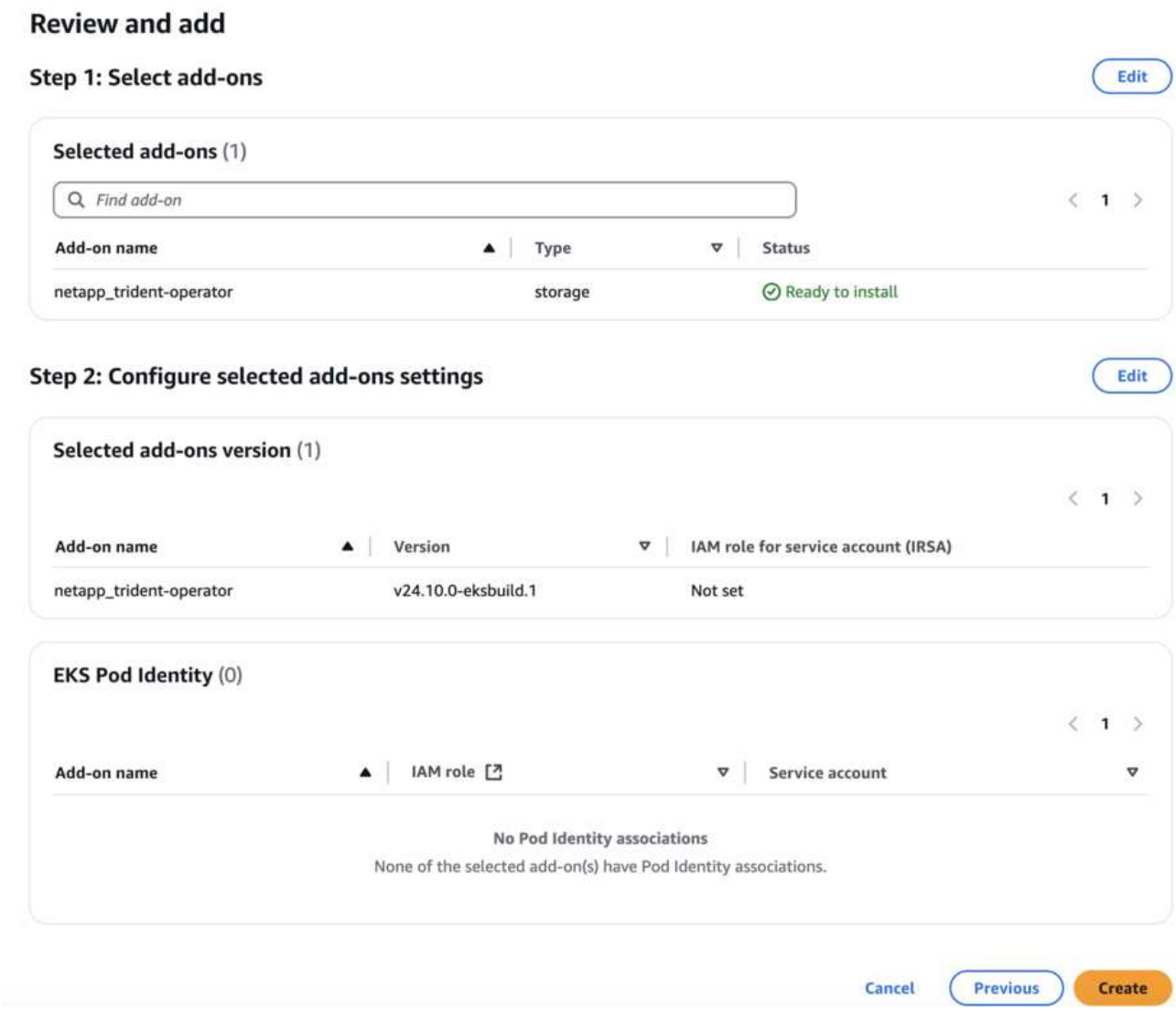
3. Accédez à **add-ons** AWS Marketplace et choisissez la catégorie *Storage*.



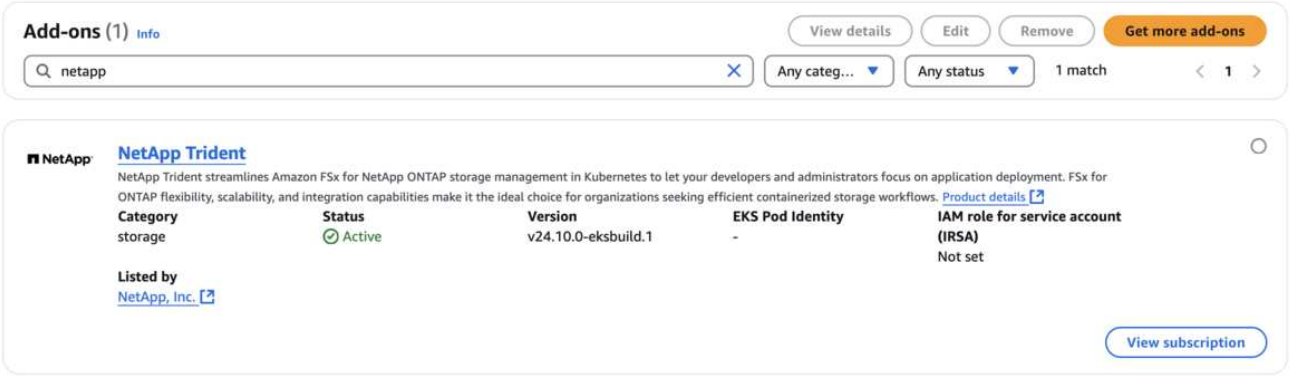
4. Localisez **NetApp Trident** et cochez la case du module complémentaire Trident, puis cliquez sur **Suivant**.
5. Choisissez la version souhaitée du module complémentaire.



6. Configurez les paramètres du module complémentaire requis.



7. Si vous utilisez IRSA (rôles IAM pour le compte de service), reportez-vous aux étapes de configuration supplémentaires"ici" .
8. Sélectionnez **Créer**.
9. Vérifiez que l'état du complément est *Active*.



10. Exécutez la commande suivante pour vérifier que Trident est correctement installé sur le cluster :

```
kubectl get pods -n trident
```

11. Poursuivez l'installation et configurez le stockage backend. Pour plus d'informations, consultez "Configurez le back-end de stockage".

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

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

L'exemple de commande suivant installe le module complémentaire Trident EKS : `eksctl create addon --cluster clusterName --name netapp_trident-operator --version v25.6.0-eksbuild.1` (avec une version dédiée)

L'exemple de commande suivant installe le module complémentaire Trident EKS version 25.6.1 : `eksctl create addon --cluster clusterName --name netapp_trident-operator --version v25.6.1-eksbuild.1` (avec une version dédiée)

L'exemple de commande suivant installe le Trident EKS add-on version 25.6.2 : `eksctl create addon --cluster clusterName --name netapp_trident-operator --version v25.6.2-eksbuild.1` (avec une version dédiée)

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

La commande suivante désinstalle le module complémentaire 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.

AVERTISSEMENT	`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.

REMARQUE	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`.
ASTUCE	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: password
```

Ce tableau récapitule les champs à inclure dans le Secret pour chaque plate-forme de stockage :

Description des champs secrets de la plate-forme de stockage	Secret	Description des champs
Azure NetApp Files	ID client	ID client d'un enregistrement d'application
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.

AVERTISSEMENT	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
```

REMARQUE

- 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
```

REMARQUE	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 back-end continuera à exister dans un état « 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
```

REMARQUE 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: ontap-gold
provisioner: csi.trident.netapp.io
mountOptions:
  - nfsvers=3
  - nolock
parameters:
  backendType: "ontap-nas"
  media: "ssd"
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.

REMARQUE	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.

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

Présentation

Une "*PersistentVolumeClaim*" demande de volume persistant 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.

Une fois la demande de volume créée, vous pouvez la monter dans un pod.

Créer la PVC

Étapes

1. Créer la PVC.

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

2. 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    1Gi        RWO                   5m
```

1. Montez le volume dans un pod.

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

REMARQUE 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
```

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

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

Exemples de manifestes

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: storage
      persistentVolumeClaim:
       claimName: pvc-storage
  containers:
    - name: pv-container
      image: nginx
      ports:
        - containerPort: 80
          name: "http-server"
      volumeMounts:
        - mountPath: "/my/mount/path"
          name: storage
```

Configuration NVMe/TCP de base

```
apiVersion: v1
kind: Pod
metadata:
  name: pod-nginx
spec:
  volumes:
    - name: basic-pvc
      persistentVolumeClaim:
        claimName: pvc-san-nvme
  containers:
    - name: task-pv-container
      image: nginx
      volumeMounts:
        - mountPath: "/my/mount/path"
          name: basic-pvc
```

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.

Déplacer un volume

À partir de Trident 26.06, un administrateur de stockage peut déplacer sans interruption un volume iSCSI vers un autre nœud ONTAP et un autre agrégat au sein du même cluster ONTAP, tandis que la charge de travail qui l'utilise continue de fonctionner. Vous demandez le déplacement de manière déclarative avec une `TridentVolumeMove` ressource personnalisée ou l' `tridentctl` équivalent, et Trident coordonne ONTAP et l'hôte afin que les E/S ne soient jamais interrompues.

De formation

Le déplacement de volume s'applique au `ontap-san` driver uniquement pour les volumes iSCSI. Les protocoles NVMe et NAS ne sont pas pris en charge.

Avant de déplacer un volume, assurez-vous que les conditions suivantes sont remplies :

- Le backend utilise le `ontap-san`pilote avec l'API REST ONTAP.
- Le cluster ONTAP comporte au moins deux nœuds.
- Vous possédez l'un des ensembles d'identifiants suivants :
  - ONTAP 9.19.1 ou version ultérieure si vous utilisez le rôle ONTAP personnalisé dédié.
  - Identifiants d'administrateur de cluster sur les versions ONTAP antérieures.

Comment ça marche

Trident déplace le volume (LUN) entre deux emplacements au sein du même cluster ONTAP. Avant le déplacement des données, Trident ajoute de nouveaux chemins iSCSI vers la cible, en plus des chemins existants, afin que le périphérique multipath hôte ne perde jamais l'accès. Une fois le déplacement terminé par ONTAP, Trident supprime les chemins source obsolètes.

Trident gère le déplacement à travers une série d'états ordonnés visibles dans la `TridentVolumeMove` ressource, ce qui vous permet de suivre la progression du déplacement.

Déplacer un volume

Identifiez le volume et les emplacements

1. Trouvez le nom du PV à déplacer. Le nom du PV correspond également au nom de la ressource à déplacer.
2. À partir du cluster ONTAP, notez le nœud et l'agrégat actuels (source) pour ce volume, puis choisissez le nœud et l'agrégat de destination (cible).

Démarrez le déplacement

Vous pouvez utiliser l'une ou l'autre méthode. Les deux créent la même `TridentVolumeMove` ressource, dont le nom est celui du PV.

Utilisation de tridentctl

```
tridentctl create volume-move \
  --volume      pvc-8a814d62-bd58-4253-b0d1-82f2885db671 \
  --source-node ontap-node-01 \
  --source-pool aggr1_node01 \
  --target-node ontap-node-02 \
  --target-pool aggr1_node02 \
  -n trident
```

`tvm` est un alias pour `volume-move`. Ajoutez `--delete-after-success 10m` pour supprimer automatiquement la ressource après la réussite du déplacement.

Utilisation d'une ressource TridentVolumeMove

```
apiVersion: trident.netapp.io/v1
kind: TridentVolumeMove
metadata:
  name: pvc-8a814d62-bd58-4253-b0d1-82f2885db671   # the PV to move
  namespace: trident
spec:
  sourceNode: ontap-node-01
  sourcePool: aggr1_node01
  targetNode: ontap-node-02
  targetPool: aggr1_node02
  # deleteAfterSuccess: 10m   # optional
```

Appliquez la ressource :

```
kubectl apply -f tridentvolumemove.yaml
```

Champs de spécification

`sourceNode`/ `targetNode`

Noms des nœuds ONTAP (et non des UUID).

`sourcePool`/ `targetPool`

Noms d'agrégats ONTAP.

`deleteAfterSuccess`

Durée facultative, par exemple `30s` ou `10m`. Définissez-la sur `0` pour supprimer immédiatement la ressource, ou omettez-la pour conserver la ressource.

Surveillez le déplacement

Suivez la progression en utilisant le `tvm` nom court ou `tridentvolumemove` :

```
kubectl get tridentvolumemove -n trident
```

Pour consulter l'état actuel et la progression par nœud, décrivez la ressource :

```
kubectl describe tridentvolumemove <pv-name> -n trident
```

Le volume peut avoir des attachements simultanés pendant l'état `Moving`, car ONTAP peut prendre un certain temps pour effectuer le déplacement d'un volume important. Pour garantir un déplacement sans interruption, Trident assure la coordination entre le contrôleur Trident et le nœud Trident. Chaque état exécute les étapes suivantes dans l'ordre, se terminant soit par `Succeeded` soit par `Failed`.

État	Ce qui se passe
`Pending`	Trident accepte et valide la demande de déplacement et recueille les informations nécessaires avant que toute modification du stockage ne commence.
`ControllerStaging`	Trident prépare l'emplacement cible. Il associe les interfaces de stockage du nœud cible au LUN et met à jour Kubernetes afin qu'un redémarrage ultérieur du nœud le reconnecte à l'emplacement correct.
`NodeStaging`	Sur chaque hôte sur lequel le volume est monté, Trident ajoute les nouveaux chemins de stockage du nœud cible aux côtés des chemins existants, afin que les E/S du pod se poursuivent sur les deux ensembles de chemins.
`Moving`	ONTAP copie et transfère les données du volume vers l'agrégat cible. Cette phase est la plus longue et ONTAP la gère entièrement tandis que Trident interroge jusqu'à la fin de l'opération. De nouvelles publications sont autorisées pendant cette phase.
`ControllerUnstaging`	Le déplacement des données est terminé. Trident met à jour Kubernetes pour supprimer les chemins sources de l'état souhaité, puis dissocie les interfaces de stockage du nœud source du LUN.
`NodeUnstaging`	Sur chaque hôte, Trident supprime les chemins sources obsolètes. L'hôte ne conserve alors que les chemins cibles, et le périphérique multipath continue de servir les E/S sans interruption.
`Succeeded`	Le volume est entièrement transféré et en ligne sur la cible. Trident conserve la `TridentVolumeMove` ressource, ou la supprime automatiquement si vous définissez `deleteAfterSuccess`.
`Failed`	Une erreur irrécupérable a interrompu le déplacement ou empêché son démarrage. Trident réinitialise l'état du volume pour permettre la reprise des opérations normales et conserve la ressource avec un message d'erreur. Consultez le message, corrigez la cause racine, supprimez la ressource et réessayez.

La `.status.attachments`liste indique la progression de chaque nœud connecté dans

son champ `state :

État d'attachement	Signification
`Pending`	Nœud identifié comme attaché ; le travail n'a pas encore commencé.
`Bridged`	De nouveaux chemins cibles sont ajoutés en parallèle des chemins existants.
`Migrated`	Le trafic est désormais acheminé via les nouveaux chemins.
`Cleaned`	Les chemins sources obsolètes ont été supprimés du nœud.
`Detached`	Le nœud a été détaché en cours de déplacement et exclu de toute considération ultérieure pendant que Trident évalue si la connexion est prête à passer à l'étape suivante.
`Failed`	Erreur terminale sur ce nœud. Consultez le message.

Nettoyez la ressource de déplacement

Si vous définissez `deleteAfterSuccess`, Trident supprime automatiquement la ressource. Sinon, vous pouvez la supprimer manuellement. Avant de demander un autre déplacement sur le même volume, assurez-vous qu'aucune ressource `TridentVolumeMove` n'existe pour ce volume, ni dans un état `Succeeded` ni dans un état `Failed`.

REMARQUE Trident ne supprime jamais automatiquement les déplacements ayant échoué. Trident conserve une `Failed` ressource à titre d'audit. Après avoir corrigé la cause première, supprimez la ressource manuellement avant de réessayer.

Développement des volumes

Trident offre aux utilisateurs de Kubernetes la possibilité d'étendre leurs volumes après leur création. Recherchez des informations sur les configurations requises pour étendre les volumes iSCSI, NFS, SMB, NVMe/TCP et FC.

Développez un volume iSCSI

Vous pouvez développer un volume persistant iSCSI à l'aide du mécanisme de provisionnement CSI.

REMARQUE 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 FC

Vous pouvez étendre un volume persistant FC à l'aide du mécanisme de provisionnement CSI.

REMARQUE	L'extension de volume FC est prise en charge par le `ontap-san` pilote et requiert Kubernetes 1.16 et versions ultérieures.

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 FC, il existe deux scénarios :

- 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 pour les PV NFS provisionnés sur `ontap-nas` , `ontap-nas-economy` , `ontap-nas-flexgroup` , et `azure-netapp-files` backends.

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
```

```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 doit créer un PV NFS de 20 Mio pour ce PVC :

```
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 PV de 20 Mio nouvellement créé à 1 Gio, modifiez le PVC et définissez `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    |
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
```

Comprendre les limites du sous-système RWX NVMe

ReadWriteMany (RWX) volumes utilisant le protocole NVMe ont une limite de scalabilité de 64 nœuds par volume. Ce qui suit inclut les limitations, explique l'architecture du sous-système NVMe concernée et décrit les étapes de résolution requises.

Comprendre la limite de 64 nœuds

Si vous prévoyez d'utiliser des volumes ReadWriteMany (RWX) avec le protocole NVMe, un seul volume NVMe RWX ne peut pas être monté par plus de 64 nœuds dans un cluster Kubernetes.

Ne planifiez pas de charges de travail qui montent le même RWX NVMe PersistentVolumeClaim sur plus de 64 nœuds.

Cette limitation s'applique uniquement aux volumes RWX qui utilisent le protocole NVMe.

Comprendre les modèles de sous-système NVMe

Modèle de sous-système par volume (Trident releases earlier than 26.02)

Dans les versions de Trident antérieures à 26.02, les volumes NVMe RWX sont provisionnés à l'aide d'un

modèle de sous-système par volume. Chaque volume NVMe RWX est associé à son propre sous-système NVMe dédié sur ONTAP.

Ce modèle est simple, mais il a une limite d'évolutivité inférieure. Dans les grands clusters Kubernetes, les limites des contrôleurs de sous-système sont rapidement atteintes car chaque volume RWX consomme un sous-système dédié.

Modèle de super-sous-système (introduit dans Trident 26.02)

À partir de Trident 26.02, les volumes RWX NVMe utilisent un modèle de super-sous-système partagé. Plusieurs volumes RWX NVMe partagent le même sous-système NVMe.

Chaque super-sous-système prend en charge jusqu'à 1024 espaces de noms (volumes). Ce modèle améliore considérablement l'évolutivité des charges de travail RWX et réduit la probabilité d'atteindre les limites du sous-système ONTAP.

Chaque volume RWX NVMe prend en charge jusqu'à 64 nœuds.

Identifier les symptômes d'erreur

Si vous créez ou attachez des volumes RWX NVMe à grande échelle, vous pourriez observer des erreurs similaires aux suivantes :

```
Maximum number of controllers reached. No more controllers can be created.
```

Cette erreur indique que la limite du contrôleur du sous-système ONTAP NVMe a été atteinte.

Résoudre les erreurs de limite du sous-système

Pour dépasser les limitations des sous-systèmes par volume et profiter du modèle de super-sous-système, mettez à niveau vers Trident 26.02 ou une version ultérieure.

Mettre à niveau Trident pour appliquer le modèle de super-sous-système

Pour appliquer le modèle de super-sous-système aux volumes RWX NVMe :

1. Mettez à niveau Trident vers la version 26.02 ou ultérieure.
2. Réduisez à zéro tous les pods qui utilisent des volumes RWX NVMe.
3. Vérifiez qu'aucune charge de travail n'utilise activement les volumes RWX NVMe.
4. Augmentez à nouveau le nombre de pods.

Cette séquence de redémarrage garantit que les volumes RWX NVMe sont attachés à l'aide du modèle super-sous-système.

- Cette limitation s'applique uniquement aux volumes RWX qui utilisent le protocole NVMe.
- La limite de 64 nœuds s'applique par volume RWX NVMe.
- Les autres modes d'accès et les autres protocoles ne sont pas affectés.

Évolutivité du contrôleur

Trident améliore la scalabilité du contrôleur grâce à une concurrence accrue sur plusieurs pilotes de stockage. Lorsque vous activez la scalabilité du contrôleur, le contrôleur Trident traite les opérations de stockage en parallèle au lieu de les sérialiser, ce qui augmente le débit dans les environnements Kubernetes comportant de nombreuses opérations simultanées.

Avant le déploiement, déterminez quels pilotes Trident prennent en charge la mise à l'échelle du contrôleur à la disponibilité générale et quels pilotes sont disponibles en version préliminaire technique dans Trident 26.06. Cela vous aide à prendre des décisions éclairées concernant le déploiement et à gérer les risques. La mise à l'échelle du contrôleur est désactivée par défaut.

Concepts clés et définitions

Évolutivité du contrôleur

La scalabilité du contrôleur fait référence à la capacité du contrôleur Trident à traiter plusieurs opérations de stockage en parallèle, au lieu de les sérialiser derrière un seul verrou. Ces opérations comprennent la création, la suppression et le redimensionnement de volumes ; la création et la suppression d'instantanés ; la publication et la dépublication de volumes ; et la gestion du backend.

Lorsque vous activez la scalabilité du contrôleur, les opérations sur différents volumes et backends s'exécutent simultanément. Cela augmente le débit et réduit le temps d'exécution global dans les environnements comportant un grand nombre d'opérations PersistentVolumeClaim et VolumeSnapshot simultanées.

Comportement par défaut (mode série)

Par défaut, le contrôleur Trident traite les opérations une à une. Chaque requête de création, de suppression, de redimensionnement ou de capture instantanée est terminée avant que Trident ne lance la suivante. Le mode série est le mode par défaut pris en charge pour toutes les installations et ne nécessite aucune configuration.

Le mode séquentiel est suffisant pour la plupart des charges de travail. N'activez la mise à l'échelle du contrôleur que lorsque le volume d'opérations crée un engorgement en mode séquentiel.

Prise en charge de l'évolutivité du contrôleur

Trident prend en charge l'évolutivité du contrôleur à différents niveaux de maturité pour différents pilotes de stockage.

Disponibilité générale

Les pilotes suivants prennent en charge la mise à l'échelle du contrôleur à la disponibilité générale dans Trident 26.06 :

- `ontap-san`
- `ontap-nas`
- `ontap-nas-economy`
- `ontap-san-economy`
- `google-cloud-netapp-volumes`

- `azure-netapp-files`
- `solidfire-san`

REMARQUE	Les `google-cloud-netapp-volumes` et `google-cloud-netapp-volumes-san` pilotes sont différents. Trident prend uniquement en charge `google-cloud-netapp-volumes`. N'utilisez pas `google-cloud-netapp-volumes-san` dans les configurations ou exemples de backend.

Aperçu technique

Le pilote suivant prend en charge la mise à l'échelle du contrôleur en tant que fonctionnalité en avant-première technique dans Trident 26.06 :

- `asa-r2` (SAN et NVMe)

Ce pilote présente les limitations suivantes :

- La gestion de la concurrence des contrôleurs est disponible uniquement à des fins d'évaluation et de test.
- Le comportement peut changer d'une version à l'autre.
- NetApp ne recommande pas l'utilisation en environnement de production.

Comment enableConcurrency fonctionne

Lorsque vous définissez `enableConcurrency` sur `true`, Trident applique un traitement simultané à tous les backends gérés par le contrôleur. Ce paramètre s'applique à tous les backends en même temps. Vous ne pouvez pas l'activer pour des backends ou des pilotes individuels.

Chaque backend configuré doit utiliser un pilote figurant dans la liste de disponibilité générale ou de préversion technique. Si un backend utilise un pilote non pris en charge, Trident ne démarre pas et Trident n'ajoute pas de backend utilisant un pilote non pris en charge.

Pour rétablir le démarrage, supprimez ou reconfigurez tout backend utilisant un pilote non pris en charge ou désactivez la scalabilité du contrôleur.

Avant d'activer

IMPORTANT	Avant d'activer la mise à l'échelle du contrôleur, vérifiez que chaque backend configuré utilise un pilote figurant dans la liste de disponibilité générale ou de préversion technique. Si un backend utilise un pilote non pris en charge, Trident ne démarre pas après l'activation de cette fonctionnalité.

Utilisez le tableau suivant pour décider d'activer ou non la mise à l'échelle du contrôleur.

Si votre environnement	Ensuite
Utilise uniquement les pilotes pris en charge et rencontre un retard dans les opérations du contrôleur	Activez la scalabilité du contrôleur.
Utilise n'importe quel pilote non pris en charge	Conservez le mode série par défaut. N'activez pas la mise à l'échelle du contrôleur.
Gère un faible volume d'opérations sans arriérés	Conservez le mode série par défaut.

Activer la scalabilité du contrôleur

L' `enableConcurrency`option de configuration contrôle la scalabilité du contrôleur. Vous devez activer explicitement cette option lors de l'installation de Trident ou lors de la mise à jour d'un déploiement existant.

Déploiement de l'opérateur Trident

Pour activer la scalabilité du contrôleur avec l'opérateur Trident, définissez `enableConcurrency` sur `true` dans la `TridentOrchestrator` ressource personnalisée (CR).

Nouvelle installation

Créez ou modifiez le `TridentOrchestrator`CR et définissez `enableConcurrency`sur `true`:

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  namespace: trident
  enableConcurrency: true
```

Appliquer la CR :

```
kubectl apply -f tridentorchestrator_cr.yaml
```

Installation existante

Corrigez le `TridentOrchestrator` CR existant pour activer l'évolutivité du contrôleur :

```
kubectl patch torc trident --type=merge -p
'{"spec":{"enableConcurrency":true}}'
```

Vérifiez que Trident a appliqué le paramètre :

```
kubectl get torc trident -o
jsonpath='{.status.currentInstallationParams.enableConcurrency}'
```

Déploiement Helm

Pour activer la scalabilité du contrôleur avec Helm, définissez la valeur `enableConcurrency` sur `true`.

Nouvelle installation

```
helm install trident netapp-trident/trident-operator --namespace trident
--create-namespace --set enableConcurrency=true
```

Installation existante

```
helm upgrade trident netapp-trident/trident-operator --namespace trident
--set enableConcurrency=true
```

Vous pouvez également définir `enableConcurrency` sur `true` dans un fichier `values.yaml` personnalisé :

```
# values.yaml
enableConcurrency: true
```

Installez ou mettez à jour ensuite avec le fichier de valeurs :

```
helm install trident netapp-trident/trident-operator --namespace trident
--create-namespace -f values.yaml
```

déploiement tridentctl

Pour activer la scalabilité du contrôleur avec `tridentctl`, transmettez le drapeau `--enable-concurrency` lors de l'installation.

Nouvelle installation

```
tridentctl install -n trident --enable-concurrency
```

Installation existante

Pour activer la mise à l'échelle du contrôleur sur un déploiement `tridentctl` existant, désinstallez Trident et réinstallez-le avec l'indicateur :

```
tridentctl uninstall -n trident
tridentctl install -n trident --enable-concurrency
```

Vérifiez que la mise à l'échelle du contrôleur est activée

Après avoir activé la mise à l'échelle du contrôleur, vérifiez que le contrôleur Trident s'exécute avec la concurrence activée. Consultez les journaux du pod du contrôleur :

```
kubectl logs -n trident deploy/trident-controller | grep -i concurrency
```

Le résultat comprend une entrée de journal confirmant que la concurrence est activée.

Comportement de concurrence

Lorsque la mise à l'échelle du contrôleur est activée, le contrôleur Trident adopte le comportement suivant :

- Trident remplace le verrou global unique par un verrouillage précis, par ressource.
- Trident sérialise les opérations qui modifient la même ressource afin de maintenir la cohérence des données.
- Les opérations qui ne font que lire une ressource s'exécutent simultanément avec d'autres opérations de lecture sur cette ressource.
- Trident limite le nombre de requêtes API ONTAP simultanées à 20 par LIF de gestion afin d'éviter la surcharge des systèmes de stockage backend.
- Si plusieurs backends partagent le même LIF de gestion, ils partagent cette limite de 20 requêtes.

Avertissements et limitations

Les considérations suivantes s'appliquent à l'évolutivité du contrôleur dans Trident 26.06 :

- La mise à l'échelle du contrôleur ne prend en charge que les pilotes figurant dans les listes de disponibilité générale et d'aperçu technique. Pour plus de détails, consultez Avant d'activer.
- Le contrôleur Trident gère la concurrence en interne. Cette version ne propose aucune limite de concurrence configurable par l'utilisateur.
- Le débit global dépend du pilote de stockage utilisé, de la réactivité du backend et des performances du serveur API Kubernetes.
- Une forte concurrence peut augmenter la charge sur les systèmes de stockage backend.
- Le comportement de mise à l'échelle du contrôleur n'est pas identique pour tous les pilotes.
- Le pilote en préversion technique peut présenter des performances incohérentes en cas de charge élevée et son comportement peut changer d'une version à l'autre.
- Le débogage des opérations concurrentes peut s'avérer plus complexe en raison de l'exécution parallèle. Les métriques et les journaux peuvent afficher les résultats des opérations entrelacées.

Recommandations

Avant d'activer la mise à l'échelle du contrôleur, complétez les étapes suivantes :

1. Confirmez que chaque backend configuré utilise un pilote figurant dans la liste de disponibilité générale ou de préversion technique.
2. Testez la modification sur un cluster hors production avant de l'appliquer en production.
3. Vérifiez que la mise à l'échelle du contrôleur est activée après avoir appliqué la modification.

Appliquez les recommandations générales suivantes lorsque vous utilisez la mise à l'échelle du contrôleur :

- Utilisez des pilotes à disponibilité générale pour les environnements de production nécessitant une évolutivité élevée.

- Évaluez le pilote en préversion technique uniquement dans des environnements hors production.
- Surveillez les performances du backend et du contrôleur lorsque vous opérez à grande échelle.
- Ne présumez pas de l'ordre des opérations dans les scripts d'automatisation.

Extension automatique du volume

L'extension automatique des volumes permet aux Persistent Volumes provisionnés par Trident d'augmenter automatiquement leur capacité lorsque celle-ci atteint un seuil défini. Cette fonctionnalité réduit les coûts d'exploitation et contribue à prévenir les interruptions d'application dues à la saturation de la capacité. L'extension automatique du volume est mise en œuvre à l'aide des Autogrow Policies. Une Autogrow Policy définit :

- Le seuil d'utilisation qui déclenche l'expansion
- La quantité dont le volume augmente
- La taille maximale que le volume peut atteindre

REMARQUE	Les volumes augmentent automatiquement lorsque le seuil d'utilisation défini est dépassé. Les volumes ne sont jamais réduits automatiquement.

De formation

Avant de configurer l'extension automatique du volume, assurez-vous que les exigences suivantes sont remplies :

- Trident 26.02 ou version ultérieure
- Autorisations de contrôle d'accès basées sur les rôles pour créer `TridentAutogrowPolicy` des ressources personnalisées
- StorageClasses configuré avec `allowVolumeExpansion: true`
- Protocoles ONTAP pris en charge :
  - Système de fichiers réseau (NFS)
  - Internet Small Computer Systems Interface (iSCSI)
  - Non-Volatile Memory Express (NVMe)
  - Protocole Fibre Channel (FCP)

Limites

- Les volumes de blocs bruts ONTAP Non-Volatile Memory Express antérieurs à ONTAP 9.16.1 ne prennent pas en charge l'extension automatique.
- Pour les volumes de réseau de stockage, si `growthAmount` est inférieur ou égal à 50 mébioctets, Trident augmente automatiquement la valeur à 51 mébioctets avant le redimensionnement, à condition que la taille résultante ne dépasse pas `maxSize`.
- Dans les environnements brownfield, l'extension automatique peut ne pas fonctionner pour certains volumes existants en raison du comportement de migration de la publication des volumes.
- Lorsqu'un volume atteint `maxSize`, aucune expansion supplémentaire ne se produit.
- Protocoles pris en charge pour l'extension automatique du volume :

- Système de fichiers réseau (NFS)
- Internet Small Computer Systems Interface (iSCSI)
- Non-Volatile Memory Express (NVMe)
- Protocole Fibre Channel (FCP)

Provisionnement de volumes avec stratégie d'extension automatique

La politique Autogrow peut être configurée à deux niveaux :

- Niveau de classe de stockage : définit la valeur par défaut pour tous les volumes (à l'aide d'une annotation)
- Niveau PVC : Remplace la valeur par défaut de la classe de stockage (en utilisant une annotation)

Créer une politique Autogrow

Les politiques Autogrow permettent l'expansion automatique des volumes lorsque ceux-ci atteignent un seuil de capacité défini.

Assurez-vous d'avoir :

- Trident 26.02 ou version ultérieure installé
- Autorisations de contrôle d'accès basées sur les rôles pour créer `TridentAutogrowPolicy` des ressources
- Compréhension des exigences de croissance de la charge de travail

Une politique Autogrow définit comment les volumes s'étendent automatiquement lorsqu'ils atteignent un seuil de capacité défini.

Vous pouvez créer des stratégies Autogrow à tout moment de votre flux de travail :

- Avant que les StorageClasses et les volumes ne soient créés
- Après que les StorageClasses existent
- Une fois les volumes provisionnés

Cette flexibilité vous permet d'introduire une extension automatique sans recréer les ressources existantes.

Spécifications de la politique Autogrow

Les politiques d'autogrow sont des ressources personnalisées Kubernetes définies comme suit :

Champ	Description	Format	Obligatoire	Exemple	Valeur par défaut
nom	Identifiant unique de la politique	Chaîne	Oui.	production-db-policy	Aucune
usedThreshold	Pourcentage de capacité qui déclenche l'expansion	Chaîne de pourcentage	Oui.	"80%"	Aucune
growthAmount	Montant à augmenter lorsque le seuil est atteint	Pourcentage ou taille	Non	« 10% » ou « 5Gi »	"10%"

Champ	Description	Format	Obligatoire	Exemple	Valeur par défaut
maxSize	Limite de taille maximale du volume	quantité Kubernetes	Non	"500Gi"	Illimité

Créer une politique Autogrow

Étapes

1. Créez un fichier YAML qui définit votre Autogrow Policy :

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: standard-autogrow
spec:
  usedThreshold: "80%"
  growthAmount: "10%"
  maxSize: "500Gi"
```

2. Appliquez la politique à votre cluster :

```
kubectl apply -f autogrow-policy.yaml
```

3. Vérifiez que la policy a été créée :

```
kubectl get tridentautogrowpolicy standard-autogrow
```

Résultat attendu

```
NAME                USED THRESHOLD   GROWTH AMOUNT   STATE
standard-autogrow   80%              10%             Success
```

États de la stratégie

Après avoir créé une politique, Trident valide la spécification et lui attribue l'un des états suivants :

État	Description	Action requise
Succès	La stratégie est validée et prête à l'emploi.	Aucun.
Échec	Des erreurs de validation ont été détectées.	Examinez et corrigez la spécification.
Suppression	Suppression en cours.	Attendez la fin.

Associer une politique à une StorageClass

Vous pouvez associer une stratégie d'autogrow à un StorageClass en utilisant l'annotation `trident.netapp.io/autogrowPolicy`. Tous les volumes provisionnés à partir de ce StorageClass héritent de la stratégie.

REMARQUE	Le StorageClass doit avoir `allowVolumeExpansion: true`.

Étapes

1. Créez ou modifiez un StorageClass avec l'annotation de stratégie d'extension automatique :

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-gold
  annotations:
    trident.netapp.io/autogrowPolicy: "production-db-policy"
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
  fsType: "ext4"
allowVolumeExpansion: true
```

2. Appliquez le StorageClass :

```
kubectl apply -f storageclass.yaml
```

3. Vérifiez l'annotation :

```
kubectl get storageclass ontap-gold -o
jsonpath='{.metadata.annotations.trident\.netapp\.io/autogrowPolicy}'
```

Résultat attendu

```
production-db-policy
```

Précédence de la politique

Lorsque des annotations de stratégie d'extension automatique sont définies à la fois sur un StorageClass et un PVC, Trident applique les règles de priorité suivantes :

1. **L'annotation PVC est prioritaire.** Si un PVC définit `trident.netapp.io/autogrowPolicy`, cette valeur est toujours utilisée.
2. **StorageClass annotation s'applique uniquement lorsque le PVC ne comporte aucune annotation.**

3. **Si aucune des deux ne comporte l'annotation,** aucune Autogrow Policy n'est appliquée.

Annotation StorageClass	Annotation PVC	Comportement efficace
`trident.netapp.io/autogrowPolicy: standard-agp`	Non défini	Utilisations `standard-agp`.
`trident.netapp.io/autogrowPolicy: standard-agp`	`trident.netapp.io/autogrowPolicy: logs-policy`	Utilise `logs-policy` (les PVC remplacent StorageClass).
`trident.netapp.io/autogrowPolicy: standard-agp`	`trident.netapp.io/autogrowPolicy: "none"`	Aucune politique d'autogrow (PVC désactive l'autogrow).
Non défini	`trident.netapp.io/autogrowPolicy: dev-policy`	Utilisations `dev-policy`.
Non défini	Non défini	Aucune politique d'autocroissance.

Exemples de configuration

Les exemples suivants montrent les configurations courantes d'Autogrow Policy pour différents cas d'usage.

Politique conservatrice pour les bases de données de production

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: production-db-policy
spec:
  usedThreshold: "75%"
  growthAmount: "20%"
  maxSize: "5Ti"
```

Stockage de journaux avec incréments de croissance fixes

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: log-storage-policy
spec:
  usedThreshold: "90%"
  growthAmount: "10Gi"
  maxSize: "100Gi"
```

Politique minimale avec valeurs par défaut

Lorsque vous omettez `growthAmount` et `maxSize`, Trident utilise les valeurs par défaut (`10%` (croissance, taille illimitée) :

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: logs-policy
spec:
  usedThreshold: "85%"
```

Stratégie avec une valeur personnalisée maxSize et la valeur par défaut growthAmount

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: default-ga-policy
spec:
  usedThreshold: "70%"
  maxSize: "100Gi"
```

Croissance agressive avec maxSize illimité

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: aggressive-growth-policy
spec:
  usedThreshold: "80%"
  growthAmount: "150%"
```

Politique avec des pourcentages fractionnaires

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: precise-policy
spec:
  usedThreshold: "80.28%"
  growthAmount: "10.65%"
  maxSize: "100Gi"
```

REMARQUE	Les pourcentages fractionnaires sont pris en charge. Si vous spécifiez plus de trois décimales, Trident arrondit la valeur à trois décimales.

NAS StorageClass avec Autogrow

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-nas-autogrow
  annotations:
    trident.netapp.io/autogrowPolicy: "standard-autogrow"
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-nas"
  fsType: "ext4"
allowVolumeExpansion: true
```

SAN StorageClass avec Autogrow

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: database-storage
  annotations:
    trident.netapp.io/autogrowPolicy: "production-db-policy"
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
  fsType: "ext4"
allowVolumeExpansion: true
```

Gérer les stratégies Autogrow

Une fois les stratégies d'extension automatique créées, vous pouvez les consulter, les modifier et les supprimer. Vous pouvez également surveiller les volumes qui utilisent une stratégie donnée.

Afficher les politiques Autogrow

Lister toutes les politiques

Utilisez `kubectl` pour lister toutes les stratégies d'extension automatique de votre cluster :

```
kubectl get tridentautogrowpolicy
```

Vous pouvez également utiliser `tridentctl` :

```
tridentctl get autogrowpolicy
```

Afficher les détails de la politique

Pour consulter la spécification complète et l'état d'une politique :

```
kubectl describe tridentautogrowpolicy production-db-policy
```

Pour afficher une règle avec ses volumes associés au format YAML :

```
tridentctl get autogrowpolicy production-db-policy -o yaml
```

Mettre à jour une politique Autogrow

Vous pouvez modifier une politique existante pour changer son seuil, son taux de croissance ou sa taille maximale. Les modifications prennent effet immédiatement pour tous les volumes qui utilisent la politique.

IMPORTANT	Les modifications affectent tous les volumes utilisant cette stratégie. Dans la mesure du possible, testez les modifications d'abord dans un environnement hors production.

Étapes

1. Modifier la politique :

   ```
   kubectl edit tridentautogrowpolicy production-db-policy
   ```

2. Modifiez les `spec` champs selon vos besoins :

   ```
   spec:
     usedThreshold: "75%"     # Changed from 80%
     growthAmount: "20%"      # Changed from 10%
     maxSize: "1Ti"           # Changed from 500Gi
   ```

3. Enregistrez et quittez. Les modifications prennent effet immédiatement.

Considérations relatives à la mise à jour

- **Effet immédiat :** Tous les volumes qui utilisent cette politique adoptent les nouveaux paramètres lors de la prochaine évaluation de croissance.
- **Aucun redémarrage du volume requis :** Les modifications s'appliquent à la prochaine opération de croissance.
- **Testez d'abord :** Validez les modifications dans un environnement non production lorsque cela est

possible.

- **Communiquer les changements :** Informez les équipes lorsque vous modifiez des politiques partagées.

Supprimer une politique Autogrow

Les politiques d'extension automatique utilisent la protection du finaliseur pour empêcher toute suppression accidentelle pendant que les volumes les utilisent activement.

Étapes

1. Supprimez la règle :

```
kubectl delete tridentautogrowpolicy production-db-policy
```

2. Si des volumes utilisent encore cette stratégie, la suppression entre dans un état `Deleting`. Vérifiez quels volumes sont concernés :

```
tridentctl get autogrowpolicy production-db-policy -o yaml
```

3. Supprimez la stratégie de chaque volume concerné. Choisissez l'une des options suivantes :

   - **Option A : Désactiver explicitement la croissance automatique** en définissant l'annotation sur `"none"`:

```
kubectl annotate pvc <pvc-name> \
  trident.netapp.io/autogrowPolicy="none" \
  --overwrite
```

   - **Option B: Supprimer complètement l'annotation :**

```
kubectl annotate pvc <pvc-name> \
  trident.netapp.io/autogrowPolicy-
```

Comportement de suppression

Scénario	Comportement
Aucun volume n'utilise la politique	La règle est supprimée immédiatement.
Les volumes utilisent la politique	La stratégie entre dans `Deleting` état. Un finaliseur bloque la finalisation jusqu'à ce que tous les volumes soient supprimés.
Tous les volumes sont supprimés de la stratégie	Les finalizers sont supprimés et la policy est supprimée.

Surveiller l'utilisation de la politique Autogrow

Vérifiez les volumes à l'aide d'une règle

```
tridentctl get autogrowpolicy production-db-policy -o json | jq '.volumes'
```

Découvrez quelle politique un volume utilise

```
kubectl get pvc database-pvc -o
jsonpath='{.metadata.annotations.trident\.netapp\.io/autogrowPolicy}'
```

Surveiller les événements de stratégie

```
kubectl get events --field-selector
involvedObject.kind=TridentAutogrowPolicy
```

Protocoles pris en charge

Autogrow prend en charge les protocoles de stockage suivants :

- NFS
- ISCSI
- FCP
- NVMe

REMARQUE	Pour les volumes SAN, si la valeur configurée `growthAmount` est de 50 Mio ou moins, Trident augmente automatiquement la marge de croissance à 51 Mo pour l'opération de redimensionnement, tant que la taille résultante ne dépasse pas `maxSize`.

Backends pris en charge

Autogrow prend en charge les backends ONTAP. Autogrow prend également en charge Google Cloud NetApp Volumes (GCNV) pour les volumes NAS et SAN.

Limitations connues

- **Volumes de blocs bruts NVMe ONTAP :** Les volumes créés avec des versions d'ONTAP antérieures à 9.16.1 ne prennent pas en charge l'autogrow.
- **Volumes existants (déploiements brownfield) :** L'extension automatique peut ne pas fonctionner pour les volumes existants, même si une stratégie d'extension automatique valide est appliquée. Cette limitation est due à une migration en cours des publications de volumes. Pour confirmer que la migration est terminée, vérifiez les journaux du contrôleur Trident pour les messages `"Migration completed"`.

Foire aux questions

À quel moment Trident évalue le seuil ?

Trident surveille en permanence l'utilisation du volume. Lorsque la capacité utilisée dépasse le `usedThreshold`, Trident crée une demande de redimensionnement interne et augmente le volume de la valeur configurée `growthAmount`.

Par exemple, cette politique déclenche une extension à 80% de la capacité et augmente le volume de 10% à chaque fois, jusqu'à un maximum de 500 GiB :

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: standard-autogrow
spec:
  usedThreshold: "80%"
  growthAmount: "10%"
  maxSize: "500Gi"
```

Puis-je appliquer une stratégie après que les volumes ont déjà été provisionnés ?

Oui. Vous pouvez créer une stratégie d'autogrow à tout moment et l'appliquer aux PVC existants en ajoutant ou en mettant à jour l' `trident.netapp.io/autogrowPolicy`annotation. Vous n'avez pas besoin de recréer le PVC ni le StorageClass.

Pour appliquer une politique à un PVC existant :

```
kubectl annotate pvc <pvc-name> \
  trident.netapp.io/autogrowPolicy="production-db-policy" \
  --overwrite
```

Pour appliquer une politique à un StorageClass existant :

```
kubectl annotate storageclass ontap-gold \
  trident.netapp.io/autogrowPolicy="production-db-policy" \
  --overwrite
```

Que se passe-t-il si je configure une stratégie d'autogrow à la fois sur le StorageClass et sur le PVC ?

L'annotation PVC est toujours prioritaire. Si un PVC possède l' `trident.netapp.io/autogrowPolicy`annotation, Trident utilise cette valeur, quelle que soit la valeur spécifiée par la StorageClass. Consultez "Précédence de la politique" pour plus de détails.

Par exemple, étant donné ce StorageClass :

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-gold
  annotations:
    trident.netapp.io/autogrowPolicy: "standard-agp"
provisioner: csi.trident.netapp.io
allowVolumeExpansion: true
```

Et ce PVC qui annule la politique StorageClass :

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: database-pvc
  annotations:
    trident.netapp.io/autogrowPolicy: "logs-policy"
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 50Gi
  storageClassName: ontap-gold
```

Trident utilise `logs-policy` pour `database-pvc`, pas `standard-agp`.

Comment désactiver l'autogrow pour un volume spécifique ?

Définissez l'annotation PVC sur `"none"`. Ce paramètre remplace toute stratégie de niveau StorageClass pour ce volume :

```
kubectl annotate pvc <pvc-name> \
  trident.netapp.io/autogrowPolicy="none" \
  --overwrite
```

Vous pouvez vérifier que la croissance automatique est désactivée :

```
kubectl get pvc <pvc-name> -o jsonpath
='{.metadata.annotations.trident\.netapp\.io/autogrowPolicy}'
```

Résultat attendu

```
none
```

Que se passe-t-il lorsqu'un volume atteint maxSize?

Trident cesse d'étendre le volume. Il ne crée plus de demandes de redimensionnement pour ce volume, même si l'utilisation continue d'augmenter au-delà de `usedThreshold`.

Par exemple, avec cette politique, Trident cesse d'augmenter le volume une fois qu'il atteint 100 Gio :

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: capped-policy
spec:
  usedThreshold: "90%"
  growthAmount: "10Gi"
  maxSize: "100Gi"
```

Pour autoriser une croissance illimitée, omettez `maxSize` ou définissez-la sur `0` :

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: unlimited-policy
spec:
  usedThreshold: "85%"
  growthAmount: "10%"
```

Puis-je modifier une stratégie sans redémarrer les volumes ?

Oui. Lorsque vous mettez à jour une stratégie, tous les volumes qui utilisent cette stratégie adoptent les nouveaux paramètres lors de la prochaine évaluation de la croissance. Aucun redémarrage de volume n'est requis.

Pour mettre à jour une politique en place :

```
kubectl edit tridentautogrowpolicy production-db-policy
```

Modifiez les champs si nécessaire :

```
spec:
  usedThreshold: "75%"     # Changed from 80%
  growthAmount: "20%"      # Changed from 10%
  maxSize: "1Ti"           # Changed from 500Gi
```

Enregistrez et quittez. Vérifiez la politique mise à jour :

```
kubectl get tridentautogrowpolicy production-db-policy
```

Résultat attendu

```
NAME                    USED THRESHOLD   GROWTH AMOUNT   STATE
production-db-policy    75%              20%             Success
```

Pourquoi ma stratégie est-elle en état d'échec ?

Un `Failed` état indique que la spécification de la politique contient des erreurs de validation. Exécutez la commande suivante pour afficher les détails de l'erreur :

```
kubectl describe tridentautogrowpolicy <policy-name>
```

Les causes fréquentes incluent une valeur invalide `usedThreshold` (doit être comprise entre 1 et 99 %), une `growthAmount` qui dépasse `maxSize`, ou un format de quantité Kubernetes invalide. Corrigez la spécification et réappliquez :

```
kubectl apply -f autogrow-policy.yaml
```

Pourquoi ne puis-je pas supprimer une policy ?

Les politiques utilisent une protection de finalisation. Si des volumes utilisent encore la politique, la suppression entre dans un état `Deleting` et attend que vous retiriez tous les volumes de la politique.

Identifiez les volumes concernés :

```
tridentctl get autogrowpolicy production-db-policy -o yaml
```

Supprimez ensuite l'annotation de chaque PVC :

```
# Option A: Explicitly disable autogrow
kubectl annotate pvc <pvc-name> \
  trident.netapp.io/autogrowPolicy="none" \
  --overwrite

# Option B: Remove the annotation entirely
kubectl annotate pvc <pvc-name> \
  trident.netapp.io/autogrowPolicy-
```

Après que vous avez supprimé tous les volumes, Trident libère le finaliseur et supprime la stratégie.

Quels backends sont pris en charge par autogrow ?

Autogrow prend en charge les backends ONTAP. Il prend également en charge Google Cloud NetApp Volumes (GCNV) pour les volumes NAS et SAN.

La fonctionnalité d'extension automatique prend en charge les protocoles NFS, iSCSI, FCP et NVMe. Les volumes de blocs bruts NVMe nécessitent ONTAP 9.16.1 ou une version ultérieure.

Dans les déploiements brownfield, les volumes existants peuvent nécessiter la migration de la publication du volume avant que l'extension automatique ne prenne effet. Pour vérifier l'état de la migration, consultez les journaux du contrôleur Trident :

```
kubectl logs -l app=trident-controller -n trident | grep "Migration
completed"
```

Les exemples suivants de StorageClass montrent l'extension automatique configurée pour les backends NAS et SAN :

Backend NAS

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-nas-autogrow
  annotations:
    trident.netapp.io/autogrowPolicy: "standard-autogrow"
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-nas"
  fsType: "ext4"
allowVolumeExpansion: true
```

Backend SAN

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: database-storage
  annotations:
    trident.netapp.io/autogrowPolicy: "production-db-policy"
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
  fsType: "ext4"
allowVolumeExpansion: true
```

Quel est le montant minimal de croissance pour les volumes SAN ?

Pour les volumes SAN, la marge de croissance minimale effective est de 51 Mo. Si vous configurez une `growthAmount` de 50 Mio ou moins, Trident augmente automatiquement la croissance à 51 Mo pour l'opération de redimensionnement.

Par exemple, cette politique définit une `growthAmount` de `"40Mi"`, mais Trident applique une croissance de 51 Mo à tout volume SAN qui l'utilise :

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: san-minimal-policy
spec:
  usedThreshold: "85%"
  growthAmount: "40Mi"
  maxSize: "100Gi"
```

Pour éviter ce réglage automatique, définissez `growthAmount` sur une valeur supérieure à 50 Mio :

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: san-policy
spec:
  usedThreshold: "85%"
  growthAmount: "100Mi"
  maxSize: "500Gi"
```

Importer des volumes

Vous pouvez importer des volumes de stockage existants en tant que PV Kubernetes en utilisant `tridentctl import` ou en créant une revendication de volume persistant (PVC) avec des annotations d'importation Trident.

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.

AVERTISSEMENT	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 politique de récupération de la classe de stockage est `delete`, Trident supprime le volume de stockage lorsque le PV est supprimé.
- Par défaut, Trident gère le PVC et renomme le volume FlexVol ainsi que le LUN ou le qtree sur le backend. Vous pouvez passer le `--no-manage` indicateur pour importer un volume non géré et le `--no-rename` indicateur pour conserver le nom du volume.
  - `--no-manage`* - Si vous utilisez le `--no-manage` Trident n'effectue aucune opération supplémentaire sur le PVC ou le PV pendant le cycle de vie des objets. Le volume de stockage n'est pas supprimé lorsque le PV est supprimé, et d'autres opérations telles que le clonage et le redimensionnement de volume sont également ignorées.

◦ `--no-rename*` - Si vous utilisez le `--no-rename`flag, Trident conserve le nom du volume existant lors de l'importation des volumes et gère le cycle de vie des volumes. Cette option est prise en charge uniquement pour les pilotes `ontap-nas`, `ontap-san` (y compris les systèmes ASA r2), `ontap-san-economy`, et `ontap-nas-economy` (importation qtree).

ASTUCE	Ces options sont utiles si vous souhaitez utiliser Kubernetes pour les charges de travail conteneurisées, mais que vous souhaitez par ailleurs 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 importer un volume en utilisant soit `tridentctl import` soit en créant un PVC avec des annotations d'importation Trident.

REMARQUE	Si vous utilisez des annotations PVC, vous n'avez pas besoin de télécharger ou d'utiliser `tridentctl` pour importer le volume.

Utilisation de tridentctl

Étapes

1. Créez un fichier PVC (par exemple, `pvc.yaml`) qui sera utilisé pour créer le PVC. Le fichier PVC doit inclure `name`, `namespace`, `accessModes` et `storageClassName`. Vous pouvez éventuellement spécifier `unixPermissions` dans la définition de votre 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
   ```

   REMARQUE N'indiquez que les paramètres requis. Les paramètres supplémentaires tels que le nom du PV ou la taille du volume peuvent entraîner l'échec de la commande d'importation.

2. Utilisez le `tridentctl import` commande permettant de spécifier le nom du backend Trident contenant le volume et le nom qui identifie de manière unique le volume sur le stockage (par exemple : ONTAP FlexVol, Element Volume). Le `-f` Un argument est requis pour spécifier le chemin d'accès au fichier PVC.

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

Utilisation des annotations PVC

Étapes

1. Créez un fichier YAML PVC (par exemple, `pvc.yaml`) avec les annotations d'importation Trident requises. Le fichier PVC doit inclure :

   - `name` et `namespace` dans les métadonnées
   - `accessModes`, `resources.requests.storage`, et `storageClassName` dans les spécifications
   - Annotations :
     - `trident.netapp.io/importOriginalName`: Nom du volume sur le backend
     - `trident.netapp.io/importBackendUUID`: UUID du backend où le volume existe
     - `trident.netapp.io/notManaged` (*Facultatif*) : Définir sur `"true"` pour les volumes non gérés. La valeur par défaut est `"false"`.

Voici un exemple de spécification pour l'importation d'un volume géré :

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: <pvc-name>
  namespace: <namespace>
  annotations:
    trident.netapp.io/importOriginalName: "<volume-name>"
    trident.netapp.io/importBackendUUID: "<backend-uuid>"
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: <size>
  storageClassName: <storage-class-name>
```

2. Appliquez le fichier YAML PVC à votre cluster Kubernetes :

```
kubectl apply -f <pvc-file>.yaml
```

Trident importera automatiquement le volume et le liera au PVC.

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`.

REMARQUE

- Le `ontap-nas-economy` pilote gère le stockage au niveau du qtree, il ne prend donc pas en charge l'importation d'un volume FlexVol seul. Pour importer des qtrees avec le `ontap-nas-economy` pilote, consultez Importer des qtrees (Tech Preview). Lorsque Trident importe un qtree, il importe également le volume parent FlexVol en tant que bucket contenant ce qtree.
- Le `ontap-nas` et `ontap-nas-flexgroup` les pilotes n'autorisent pas les noms de volumes dupliqués.

Chaque volume créé avec le `ontap-nas` pilote est un FlexVol volume sur le cluster ONTAP. L'importation de volumes FlexVol avec le `ontap-nas` pilote fonctionne de la même manière. Les volumes FlexVol qui existent déjà sur un cluster ONTAP peuvent être importés en tant que `ontap-nas` demande de volume persistant. De même, les volumes FlexGroup peuvent être importés en tant que `ontap-nas-flexgroup` ESV.

Exemples ONTAP NAS utilisant tridentctl

Les exemples suivants montrent comment importer des volumes gérés et non gérés à l'aide de `tridentctl`.

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

Exemples ONTAP NAS utilisant des annotations PVC

Les exemples suivants montrent comment importer des volumes gérés et non gérés à l'aide d'annotations PVC.

Volume géré

L'exemple suivant importe un volume de 1 GiB `ontap-nas` nommé `ontap_volume1` à partir du backend `81abcb27-ea63-49bb-b606-0a5315ac5f21` avec le mode d'accès RWO défini à l'aide d'annotations PVC :

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: <managed-imported-volume>
  namespace: <namespace>
  annotations:
    trident.netapp.io/importOriginalName: "ontap_volume1"
    trident.netapp.io/importBackendUUID: "81abcb27-ea63-49bb-b606-
0a5315ac5f21"
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: <storage-class-name>
```

Volume non géré

L'exemple suivant importe 1Gi `ontap-nas` volume nommé `ontap-volume2` depuis le backend `34abcb27-ea63-49bb-b606-0a5315ac5f34` avec le mode d'accès RWO défini à l'aide d'annotations PVC :

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: <umanaged-imported-volume>
  namespace: <namespace>
  annotations:
    trident.netapp.io/importOriginalName: "ontap-volume2"
    trident.netapp.io/importBackendUUID: "34abcb27-ea63-49bb-b606-
0a5315ac5f34"
    trident.netapp.io/notManaged: "true"
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: <storage-class-name>
```

Importer des qtrees

REMARQUE	L'importation Qtree est une fonctionnalité en avant-première technique de Trident 26.06. Les fonctionnalités en avant-première technique ne sont pas prises en charge pour une utilisation en production.

À partir de Trident 26.06, vous pouvez importer des qtrees ONTAP existants dans Trident à l'aide du `ontap-nas-economy` driver. Identifiez le qtree par son volume FlexVol et le nom du qtree au format `flexvol/qtree`, ainsi que par l'UUID du backend qui le contient. Les considérations générales d'importation dans Présentation et considérations s'appliquent également à l'importation de qtrees.

Importer un qtree à l'aide de tridentctl

Spécifiez le qtree au format `flexvol/qtree` comme nom du volume. L'exemple suivant importe le qtree `qtree1` du volume FlexVol `flexvol1` sur le backend `ontap_nas_economy` :

```
tridentctl import volume ontap_nas_economy flexvol1/qtree1 -f <path-to-pvc-file>
```

Pour importer le qtree sans le renommer, ajoutez l `--no-rename`option. Pour importer le qtree sans que Trident gère son cycle de vie, ajoutez l `--no-manage`option.

Importer un qtree à l'aide d'annotations PVC

Créez un PVC avec les annotations suivantes :

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: <pvc-name>
  namespace: <namespace>
  annotations:
    trident.netapp.io/importOriginalName: "flexvol1/qtree1"
    trident.netapp.io/importBackendUUID: "<backend-uuid>"
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: <size>
  storageClassName: <storage-class-name>
```

Pour conserver les noms de volume FlexVol et de qtree existants, ajoutez l' `trident.netapp.io/importNoRename: "true"`annotation. Pour importer le qtree comme non géré, ajoutez l' `trident.netapp.io/notManaged: "true"`annotation.

Comportement de dénomination et de gestion

Importation gérée (par défaut)

Trident renomme le qtree pour qu'il corresponde au nom du PVC et renomme le volume FlexVol selon la convention d'appellation `ontap-nas-economy`. Le chemin de jonction ne change pas.

`--no-rename`

Trident conserve les noms de volumes FlexVol® et de qtrees existants. Les noms doivent déjà respecter les conventions de nommage de Trident, sinon Trident ne peut pas gérer les opérations de suppression ou de montage pour le qtree.

`--no-manage`

Trident fait référence au qtree mais n'en gère pas le cycle de vie. Les noms doivent respecter les conventions d'appellation de Trident, et Trident n'utilise ni ne gère les règles d'export automatiques pour le qtree.

Comportement des règles d'export

Si le backend utilise des règles d'export automatiques, le qtree importé utilise le modèle de règles d'export par qtree, dans lequel chaque qtree possède sa propre règle d'export. Lors de l'importation, Trident attribue au qtree une règle d'export vide. Au moment de la publication, Trident met à jour le volume FlexVol avec la règle d'export du backend si l'export automatique est activé et si la règle d'export FlexVol existante est `default` ou vide.

AVERTISSEMENT	La mise à jour des règles d'export du volume FlexVol peut perturber les autres qtrees du même volume FlexVol qui sont actuellement montés. Importez les qtrees uniquement lorsque les qtrees du volume FlexVol sont démontés.
AVERTISSEMENT	Ne faites pas pointer deux clusters Kubernetes vers la même SVM active et n'importez pas un qtree déjà géré par un autre cluster. Si les deux clusters gèrent le même volume, des opérations conflictuelles peuvent corrompre ou détruire les données.

SAN ONTAP

Trident prend en charge l'importation de volumes à l'aide du `ontap-san` (iSCSI, NVMe/TCP et FC) et `ontap-san-economy` conducteurs.

Trident peut importer des volumes ONTAP SAN FlexVol contenant un seul LUN. Ceci est cohérent avec le `ontap-san` pilote, qui crée un FlexVol volume pour chaque PVC et un LUN dans le FlexVol volume. Trident importe le FlexVol volume et l'associe à la définition PVC. Trident peut importer `ontap-san-economy` volumes contenant plusieurs LUN.

Les exemples suivants montrent comment importer des volumes gérés et non gérés :

Volume géré

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

L'exemple suivant importe le `ontap-san-managed` FlexVol volume 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
```

Exemples d'économie SAN ONTAP

Les exemples suivants montrent comment importer des volumes gérés et non gérés pour le backend `ontap-san-economy`.

Volume géré

Lors de l'importation d'un volume géré, Trident prend possession du FlexVol et le renomme. Vous devez tenir compte de ce renommage lors de l'importation de plusieurs LUN provenant du même FlexVol.

L'exemple suivant importe `lun1` du FlexVol `toimport` en tant que volume géré nommé `vol-managed-saneco` :

```
tridentctl import volume vol-managed-saneco toimport/lun1 -f
import1.yaml
```

Après l'importation `lun1`, Trident renomme le FlexVol (par exemple, en `trident_lun_pool_xyz`). Pour importer d'autres LUNs du même FlexVol, utilisez le nouveau nom FlexVol :

```
tridentctl import volume vol-managed-saneco trident_lun_pool_xyz/lun2
-f import2.yaml
```

REMARQUE	Le `ontap-san-economy` backend importe une LUN à la fois. Vous pouvez automatiser plusieurs importations à l'aide d'un script.

Volume non géré

Lorsque vous importez un volume non géré, Trident ne prend pas possession du FlexVol®. Cependant, le FlexVol® et la LUN doivent suivre les conventions de nommage Trident.

Format de nommage FlexVol®

```
trident_lun_pool_STORAGEPREFIX_RANDOMSTRING
```

- `STORAGEPREFIX` est la valeur de `storagePrefix` dans votre configuration backend. La valeur par défaut est `trident`.
- `RANDOMSTRING` est n'importe quelle chaîne de caractères que vous choisissez.

Exigence de dénomination des LUN

Le LUN doit être nommé `lun0`.

Exemple

Si votre `storagePrefix` est `xyz`, le chemin complet vers le LUN est :

```
trident_lun_pool_xyz_randomstring/lun0
```

Elément

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

REMARQUE	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.

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

Azure NetApp Files

Trident prend en charge l'importation de volumes à l'aide du `azure-netapp-files` pilote.

REMARQUE	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`.

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

Google Cloud NetApp volumes

Trident prend en charge l'importation de volumes à l'aide du `google-cloud-netapp-volumes` pilote.

L'exemple suivant importe un volume sur le backend `backend-tbc-gcnv1` avec le volume `testvoleasiaeast1`.

```
tridentctl import volume backend-tbc-gcnv1 "testvoleasiaeast1" -f < path-
to-pvc> -n trident

+----------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
|                   NAME                 |  SIZE   | STORAGE CLASS
| PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+----------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
| pvc-a69cda19-218c-4ca9-a941-aea05dd13dc0 |  10 GiB | gcnv-nfs-sc-
identity | file     | 8c18cdf1-0770-4bc0-bcc5-c6295fe6d837 | online | true
|
+----------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
```

L'exemple suivant importe un `google-cloud-netapp-volumes` volume lorsque deux volumes sont présents dans la même région :

```
tridentctl import volume backend-tbc-gcnv1
"projects/123456789100/locations/asia-east1-a/volumes/testvoleasiaeast1"
-f <path-to-pvc> -n trident

+------------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
|                   NAME                   |  SIZE   | STORAGE CLASS
| PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+------------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
| pvc-a69cda19-218c-4ca9-a941-aea05dd13dc0 |  10 GiB | gcnv-nfs-sc-
identity | file     | 8c18cdf1-0770-4bc0-bcc5-c6295fe6d837 | 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 :

- Opérations de création, d'importation et de clonage de volumes.
- Dans le cas du `ontap-nas-economy` pilote, seul le nom du volume Qtree est conforme au modèle de nom.
- Dans le cas du `ontap-san-economy` pilote, seul le nom du LUN est conforme au modèle de nom.

Limites

- Les noms de volumes personnalisés sont compatibles uniquement avec les pilotes ONTAP sur site.
- Les étiquettes personnalisées ne sont prises en charge que pour les `ontap-san`, `ontap-nas`, et `ontap-nas-flexgroup` conducteurs.
- Les noms de volumes personnalisés ne s'appliquent pas aux volumes existants.

Comportements clés des noms de volume personnalisables

- 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.
- 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

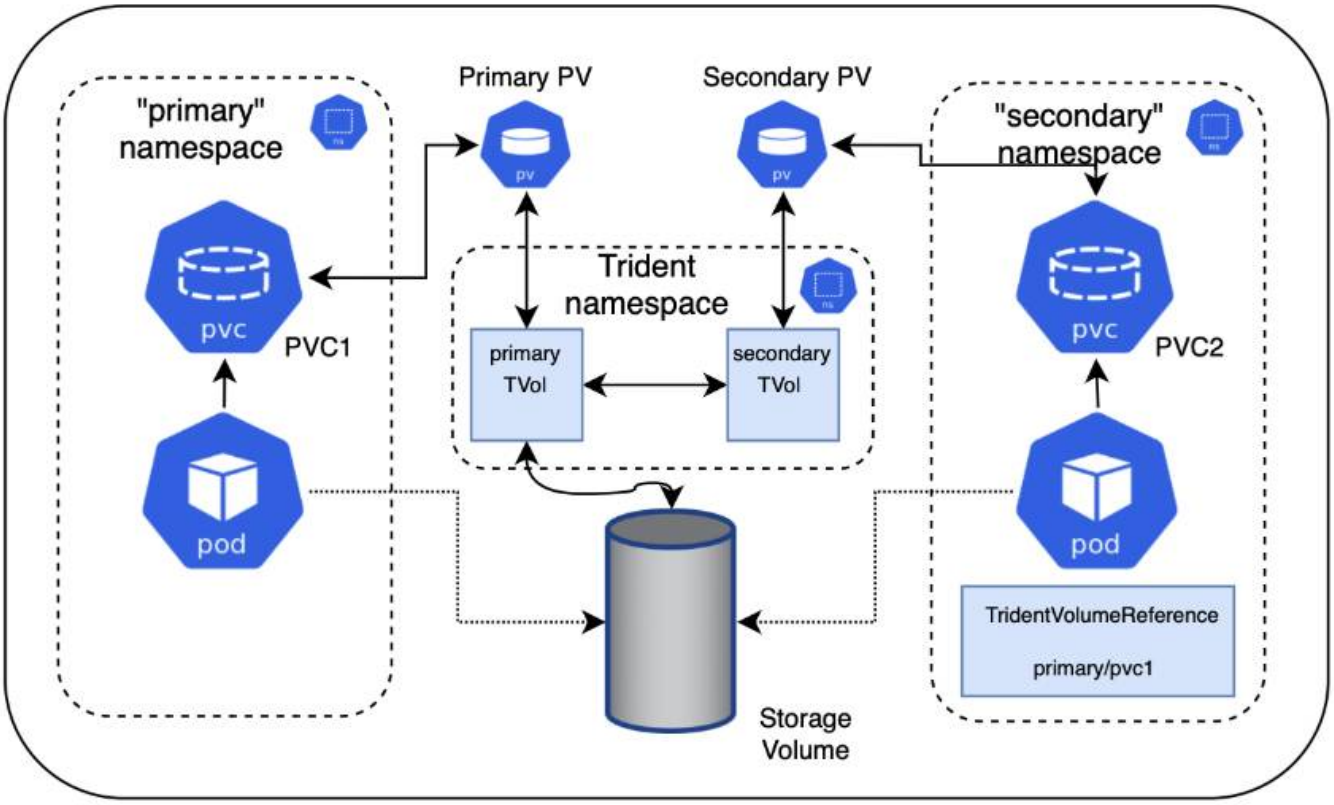
Avec Trident, vous pouvez créer un volume dans un espace de noms principal et le partager dans un ou plusieurs espaces de noms secondaires.

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.

   REMARQUE

   - 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. **Administrateur de cluster :** assurez-vous qu'un RBAC approprié est en place 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
```

REMARQUE 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

En utilisant l'[`tridentctl`utilitaire, vous pouvez exécuter la commande `get`pour obtenir les volumes subordonnés. Pour plus d'informations, consultez le lien :

../trident-reference/trident-cl.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 :

- Regardez la démo sur "NetAppTV".

Cloner des volumes entre des espaces de noms

Avec Trident, vous pouvez créer de nouveaux volumes à l'aide de volumes ou de volumes namesnapshots existants à partir d'un autre namespace situé dans le même cluster Kubernetes.

Prérequis

Avant de cloner des volumes, assurez-vous que les systèmes back-end source et de destination sont du même type et ont la même classe de stockage.

REMARQUE	Le clonage entre les espaces de noms est pris en charge uniquement pour le `ontap-san` et `ontap-nas` pilotes de stockage. Les clones en lecture seule ne sont pas pris en charge.

Démarrage rapide

Vous pouvez configurer le clonage de volumes en quelques étapes seulement.

1. **Configurez la demande de volume persistant source pour cloner 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 **Créer la demande de volume persistant de clone dans le namespace de destination**

Le propriétaire de l'espace de noms de destination crée une demande de volume persistant pour cloner la demande de volume persistant à partir de l'espace de noms source.

Configurer les espaces de noms source et de destination

Pour garantir la sécurité, le clonage de volumes entre espaces de noms nécessite la collaboration et l'action du propriétaire de l'espace de noms source, de l'administrateur du 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 (`namespace1`) (`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' `cloneToNamespace`annotation.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc1
  namespace: namespace1
  annotations:
    trident.netapp.io/cloneToNamespace: namespace2
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: trident-csi
  resources:
    requests:
      storage: 100Gi
```

Trident crée le volume PV et son volume de stockage back-end.

REMARQUE

- 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/cloneToNamespace: namespace2,namespace3,namespace4`, .
- Vous pouvez partager sur tous les espaces de noms à l'aide de `*`. Par exemple, `trident.netapp.io/cloneToNamespace: *`
- Vous pouvez mettre à jour la demande de volume persistant pour inclure l' `cloneToNamespace`annotation à tout moment.

2. **Administrateur du cluster :** Assurez-vous que le RBAC approprié est en place 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(`namespace2`).
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 :** Créez un PVC (`namespace2`)(`pvc2` dans l'espace de noms de destination en utilisant le `cloneFromPVC` ou `cloneFromSnapshot`, et les `cloneFromNamespace` annotations pour désigner le PVC source.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  annotations:
    trident.netapp.io/cloneFromPVC: pvc1
    trident.netapp.io/cloneFromNamespace: namespace1
  name: pvc2
  namespace: namespace2
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: trident-csi
  resources:
    requests:
      storage: 100Gi
```

Limites

- Pour les demandes de provisionnement virtuels provisionnées à l'aide de pilotes ONTAP-nas-Economy, les clones en lecture seule ne sont pas pris en charge.

Réplication de volumes à l'aide de SnapMirror

Trident prend en charge les relations de mise en miroir entre un volume source sur un cluster et le volume de destination sur le cluster peering pour la réplication des données à des fins de reprise après incident. Vous pouvez utiliser une définition de ressource personnalisée (CRD) avec espace de noms, appelée relation miroir Trident (TMR), pour effectuer les opérations suivantes :

- Création de relations de symétrie entre les volumes (ESV)
- Supprimez les relations de symétrie entre les volumes
- Rompez les relations de symétrie
- Promotion du volume secondaire en cas d'incident (basculements)
- Transition sans perte des applications d'un cluster à un autre (en cas de basculements ou de migrations planifiés)

Conditions préalables à la réplication

Assurez-vous que les conditions préalables suivantes sont remplies avant de commencer :

Clusters ONTAP

- **Trident** : Trident version 22.10 ou ultérieure doit exister sur les clusters Kubernetes source et destination qui utilisent ONTAP en tant que back-end.
- **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" .

  À partir de ONTAP 9.10.1, toutes les licences sont livrées sous forme de fichier de licence NetApp (NLF), qui est un fichier unique qui active plusieurs fonctionnalités. Pour plus d'informations, reportez-vous à la section "Licences incluses avec ONTAP One" .

REMARQUE	Seule la protection asynchrone SnapMirror est prise en charge.

Peering

- **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" .

IMPORTANT	S'assurer que les noms de SVM utilisés dans la relation de réplication entre deux clusters ONTAP sont uniques.

- **Trident et SVM** : les SVM distants peering doivent être disponibles pour Trident sur le cluster destination.

Pilotes pris en charge

NetApp Trident prend en charge la réplication de volumes avec la technologie NetApp SnapMirror à l'aide de classes de stockage prises en charge par les pilotes suivants : **`ontap-nas` : NFS** `ontap-san` : iSCSI

ontap-san : FC `ontap-san` : NVMe/TCP (nécessite au minimum la version ONTAP 9.15.1)

REMARQUE	La réplication de volume via SnapMirror n'est pas prise en charge sur les systèmes ASA r2. Pour plus d'informations sur les systèmes ASA r2, consultez la section "En savoir plus sur les systèmes de stockage ASA r2" .

Créer une demande de volume persistant en miroir

Suivez ces étapes et utilisez les exemples CRD pour créer une relation miroir entre les volumes principal et secondaire.

Étapes

1. Effectuez les étapes suivantes sur le cluster Kubernetes principal :
   a. Créez un objet StorageClass avec le `trident.netapp.io/replication: true` paramètre.

      Exemple

      ```
      apiVersion: storage.k8s.io/v1
      kind: StorageClass
      metadata:
        name: csi-nas
      provisioner: csi.trident.netapp.io
      parameters:
        backendType: "ontap-nas"
        fsType: "nfs"
        trident.netapp.io/replication: "true"
      ```

   b. Créez une demande de volume persistant avec une classe de stockage précédemment créée.

      Exemple

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

   c. Créez une demande de modification MirrorRelationship avec des informations locales.

Exemple

```
kind: TridentMirrorRelationship
apiVersion: trident.netapp.io/v1
metadata:
  name: csi-nas
spec:
  state: promoted
  volumeMappings:
  - localPVCName: csi-nas
```

Trident récupère les informations internes du volume et l'état de protection des données (DP) actuel du volume, puis remplit le champ d'état de MirrorRelationship.

d. Obtenir le CR TridentMirrorRelationship pour obtenir le nom interne et la SVM du PVC.

```
kubectl get tmr csi-nas
```

```
kind: TridentMirrorRelationship
apiVersion: trident.netapp.io/v1
metadata:
  name: csi-nas
  generation: 1
spec:
  state: promoted
  volumeMappings:
  - localPVCName: csi-nas
status:
  conditions:
  - state: promoted
    localVolumeHandle:
"datavserver:trident_pvc_3bedd23c_46a8_4384_b12b_3c38b313c1e1"
    localPVCName: csi-nas
    observedGeneration: 1
```

2. Effectuez les étapes suivantes sur le cluster Kubernetes secondaire :

   a. Créez une classe de stockage avec le paramètre trident.netapp.io/replication: true.

Exemple

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: csi-nas
provisioner: csi.trident.netapp.io
parameters:
  trident.netapp.io/replication: true
```

b. Créez une demande de modification MirrorRelationship avec les informations de destination et de source.

Exemple

```
kind: TridentMirrorRelationship
apiVersion: trident.netapp.io/v1
metadata:
  name: csi-nas
spec:
  state: established
  volumeMappings:
  - localPVCName: csi-nas
    remoteVolumeHandle:
"datavserver:trident_pvc_3bedd23c_46a8_4384_b12b_3c38b313c1e1"
```

Trident crée une relation SnapMirror avec le nom de la stratégie de relation configurée (ou par défaut pour ONTAP) et l'initialise.

c. Créez une demande de volume persistant avec une classe de stockage précédemment créée pour agir en tant que classe secondaire (destination SnapMirror).

Exemple

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: csi-nas
  annotations:
    trident.netapp.io/mirrorRelationship: csi-nas
spec:
  accessModes:
  - ReadWriteMany
resources:
  requests:
    storage: 1Gi
storageClassName: csi-nas
```

Trident vérifie la CRD TridentMirrorRelationship et ne crée pas le volume si la relation n'existe pas. Si la relation existe, Trident s'assurera que le nouveau FlexVol volume est placé sur un SVM peering avec le SVM distant défini dans le MirrorRelationship.

États de réplication des volumes

Une relation de miroir Trident (TMR) est une relation CRD qui représente une extrémité d'une relation de réplication entre les ESV. La TMR de destination a un état qui indique à Trident quel est l'état souhaité. La TMR de destination a les États suivants :

- **Établi** : le PVC local est le volume de destination d'une relation miroir, et il s'agit d'une nouvelle relation.
- **Promu**: Le PVC local est ReadWrite et montable, sans relation de miroir actuellement en vigueur.
- **Rétabli**: Le PVC local est le volume de destination d'une relation miroir et était également auparavant dans cette relation miroir.
  - L'état rétabli doit être utilisé si le volume de destination était déjà en relation avec le volume source car il écrase le contenu du volume de destination.
  - L'état rétabli échouera si le volume n'était pas auparavant dans une relation avec la source.

Promotion de la demande de volume persistant secondaire en cas de basculement non planifié

Effectuez l'étape suivante sur le cluster Kubernetes secondaire :

- Mettez à jour le champ *spec.state* de TridentMirrorRelationship vers `promoted`.

Promotion de la demande de volume persistant secondaire lors d'un basculement planifié

Lors d'un basculement planifié (migration), effectuez les étapes suivantes pour promouvoir la demande de volume persistant secondaire :

Étapes

1. Sur le cluster Kubernetes principal, créez un snapshot de la demande de volume persistant et attendez que le snapshot soit créé.

2. Sur le cluster Kubernetes principal, créez la CR SnapshotInfo pour obtenir des informations internes.

   Exemple

   ```
   kind: SnapshotInfo
   apiVersion: trident.netapp.io/v1
   metadata:
     name: csi-nas
   spec:
     snapshot-name: csi-nas-snapshot
   ```

3. Sur le cluster Kubernetes secondaire, mettez à jour le champ *spec.state* du *TridentMirrorRelationship* CR en *promu* et *spec.promotedSnapshotHandle* en tant que nom interne du snapshot.
4. Sur le cluster Kubernetes secondaire, confirmez l'état (champ status.state) de TridentMirrorRelationship à promu.

Restaurer une relation de miroir après un basculement

Avant de restaurer une relation de symétrie, choisissez le côté que vous voulez faire comme nouveau principal.

Étapes

1. Sur le cluster Kubernetes secondaire, assurez-vous que les valeurs du champ *spec.remoteVolumeHandle* du champ TridentMirrorRelationship sont mises à jour.
2. Sur le cluster Kubernetes secondaire, mettez à jour le champ *spec.mirror* de TridentMirrorRelationship sur `reestablished`.

Opérations supplémentaires

Trident prend en charge les opérations suivantes sur les volumes principal et secondaire :

Répliquer la demande de volume persistant primaire sur une nouvelle demande de volume secondaire

Assurez-vous que vous avez déjà un PVC primaire et un PVC secondaire.

Étapes

1. Supprimez les CRD PersistentVolumeClaim et TridentMirrorRelationship du cluster secondaire (destination) établi.
2. Supprimez le CRD TridentMirrorRelationship du cluster principal (source).
3. Créez un nouveau CRD TridentMirrorRelationship sur le cluster principal (source) pour le nouveau PVC secondaire (destination) que vous souhaitez établir.

Redimensionner une PVC en miroir, principale ou secondaire

La demande de volume persistant peut être redimensionnée normalement, ONTAP étendra automatiquement les flevxols de destination si la quantité de données dépasse la taille actuelle.

Supprimer la réplication d'une demande de volume persistant

Pour supprimer la réplication, effectuez l'une des opérations suivantes sur le volume secondaire actuel :

- Supprimez MirrorRelationship sur le PVC secondaire. Cela interrompt la relation de réplication.
- Ou, mettez à jour le champ spec.state à *promu*.

Suppression d'une demande de volume persistant (qui était auparavant mise en miroir)

Trident recherche les ESV répliquées et libère la relation de réplication avant toute tentative de suppression du volume.

Supprimer une TMR

La suppression d'une TMR d'un côté d'une relation symétrique entraîne la transition de la TMR restante vers l'état *promu* avant que Trident ne termine la suppression. Si la TMR sélectionnée pour la suppression est déjà à l'état *promoted*, il n'y a pas de relation miroir existante et la TMR sera supprimée et Trident promouvra la PVC locale en *ReadWrite*. Cette suppression libère les métadonnées SnapMirror pour le volume local dans ONTAP. Si ce volume est utilisé dans une relation miroir à l'avenir, il doit utiliser une nouvelle TMR avec un état de réplication *établi* volume lors de la création de la nouvelle relation miroir.

Mettre à jour les relations miroir lorsque ONTAP est en ligne

Les relations miroir peuvent être mises à jour à tout moment après leur établissement. Vous pouvez utiliser les `state: promoted` champs ou `state: reestablished` pour mettre à jour les relations. Lors de la promotion d'un volume de destination en volume ReadWrite standard, vous pouvez utiliser *promotedSnapshotHandle* pour spécifier un snapshot spécifique dans lequel restaurer le volume actuel.

Mettre à jour les relations en miroir lorsque ONTAP est hors ligne

Vous pouvez utiliser un CRD pour effectuer une mise à jour SnapMirror sans que Trident ne dispose d'une connectivité directe au cluster ONTAP. Reportez-vous à l'exemple de format de TridentActionMirrorUpdate suivant :

Exemple

```
apiVersion: trident.netapp.io/v1
kind: TridentActionMirrorUpdate
metadata:
  name: update-mirror-b
spec:
  snapshotHandle: "pvc-1234/snapshot-1234"
  tridentMirrorRelationshipName: mirror-b
```

`status.state` Reflète l'état du CRD TridentActionMirrorUpdate. Il peut prendre une valeur de *succeed*, *In Progress* ou *FAILED*.

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.

ASTUCE 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.

REMARQUE 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

```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

```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"
    }
  ]
}
```

REMARQUE

`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: null
name: pvc-san
spec: null
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

La capture instantanée de volume est prise en charge par `ontap-nas` , `ontap-nas-flexgroup` , `ontap-san` , `ontap-san-economy` , `solidfire-san` , `azure-netapp-files` , et `google-cloud-netapp-volumes` conducteurs.

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.

REMARQUE	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.

AVERTISSEMENT	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.

REMARQUE	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>
  volumeSnapshotRef:
    name: import-snap
    namespace: default
```

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
```

REMARQUE	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.

Restauration de volumes sur place à partir d'un snapshot

Trident assure une restauration rapide des volumes sur place à partir d'un snapshot à l'aide du `TridentActionSnapshotRestore` système CR (TASR). Cette CR fonctionne comme une action Kubernetes impérative et ne persiste pas une fois l'opération terminée.

Trident prend en charge la restauration des instantanés sur le `ontap-san` , `ontap-san-economy` , `ontap-nas` , `ontap-nas-flexgroup` , `azure-netapp-files` , `google-cloud-netapp-volumes` , et `solidfire-san` conducteurs.

Avant de commencer

Vous devez disposer d'une demande de volume liée et d'un instantané de volume disponible.

- Vérifiez que l'état de la demande de volume persistant est lié.

```
kubectl get pvc
```

- Vérifiez que le snapshot du volume est prêt à être utilisé.

```
kubectl get vs
```

Étapes

1. Créer la CR TASR. Cet exemple crée une CR pour la PVC `pvc1` et l'instantané de volume `pvc1-snapshot`.

   REMARQUE Le CR TIR doit se trouver dans un espace de nom où le PVC et le VS existent.

```
cat tasr-pvc1-snapshot.yaml
```

```
apiVersion: trident.netapp.io/v1
kind: TridentActionSnapshotRestore
metadata:
  name: trident-snap
  namespace: trident
spec:
  pvcName: pvc1
  volumeSnapshotName: pvc1-snapshot
```

2. Appliquez la CR pour effectuer une restauration à partir de l'instantané. Cet exemple permet de restaurer des données à partir d'un snapshot `pvc1`.

```
kubectl create -f tasr-pvc1-snapshot.yaml
```

```
tridentactionsnapshotrestore.trident.netapp.io/trident-snap created
```

Résultats

Trident restaure les données à partir du snapshot. Vous pouvez vérifier l'état de la restauration des snapshots :

```
kubectl get tasr -o yaml
```

```
apiVersion: trident.netapp.io/v1
items:
- apiVersion: trident.netapp.io/v1
  kind: TridentActionSnapshotRestore
  metadata:
    creationTimestamp: "2023-04-14T00:20:33Z"
    generation: 3
    name: trident-snap
    namespace: trident
    resourceVersion: "3453847"
    uid: <uid>
  spec:
    pvcName: pvc1
    volumeSnapshotName: pvc1-snapshot
  status:
    startTime: "2023-04-14T00:20:34Z"
    completionTime: "2023-04-14T00:20:37Z"
    state: Succeeded
kind: List
metadata:
  resourceVersion: ""
```

REMARQUE

- Dans la plupart des cas, Trident ne réessaiera pas automatiquement l'opération en cas d'échec. Vous devrez effectuer à nouveau l'opération.
- Les utilisateurs Kubernetes sans accès administrateur peuvent avoir à obtenir l'autorisation de l'administrateur pour créer une CR ASR dans l'espace de noms de leur application.

Supprimez un volume persistant avec les snapshots associés

Lors de la suppression d'un volume persistant avec snapshots associés, le volume Trident correspondant est mis à jour et passe à l'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
```

REMARQUE	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"

Travailler avec des instantanés de groupes de volumes

Instantanés de groupes de volumes Kubernetes de volumes persistants (PV) : NetApp Trident permet de créer des instantanés de plusieurs volumes (un groupe d'instantanés de volumes). Ces instantanés de groupes de volumes représentent des copies de plusieurs volumes prises simultanément.

REMARQUE	VolumeGroupSnapshot est une fonctionnalité bêta de Kubernetes avec des API bêta. Kubernetes 1.32 est la version minimale requise pour VolumeGroupSnapshot.

Créer des instantanés de groupes de volumes

La prise en charge des instantanés de groupes de volumes est assurée par les pilotes de stockage suivants :

- `ontap-san`Pilote - uniquement pour les protocoles iSCSI et FC, pas pour le protocole NVMe/TCP.
- `ontap-san-economy` - uniquement pour le protocole iSCSI.
- `ontap-nas`

REMARQUE	La création d'instantanés de groupes de volumes n'est pas prise en charge pour les systèmes de stockage NetApp ASA r2 ou AFX.

Avant de commencer

- Assurez-vous que votre version de Kubernetes est K8s 1.32 ou supérieure.
- 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 externe et les CRD, reportez-vous à Déployer un contrôleur de snapshot de volume .

REMARQUE	Ne créez pas de contrôleur de snapshot si vous créez des snapshots de groupe de volumes à la demande dans un environnement GKE. GKE utilise un contrôleur de snapshot caché intégré.

- Dans le contrôleur de snapshot YAML, définissez le `CSIVolumeGroupSnapshot` Définissez la fonction « true » pour garantir que l'instantané du groupe de volumes est activé.
- Créez les classes d'instantanés de groupe de volumes requises avant de créer un instantané de groupe de volumes.
- Assurez-vous que tous les PVC/volumes se trouvent sur le même SVM pour pouvoir créer VolumeGroupSnapshot.

Étapes

- Créez un VolumeGroupSnapshotClass avant de créer un VolumeGroupSnapshot. Pour plus d'informations, reportez-vous "Classe d'instantanés de groupe de volumes"à .

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

- Créez des PVC avec les étiquettes requises à l'aide des classes de stockage existantes ou ajoutez ces étiquettes aux PVC existants.
  1. Définissez la clé et la valeur de l'étiquette en fonction de vos besoins

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc1-group-snap
  labels:
    consistentGroupSnapshot: groupA
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 100Mi
  storageClassName: sc1-1
```

- Créer un VolumeGroupSnapshot avec la même étiquette (`consistentGroupSnapshot: groupA`) spécifié dans le PVC.

  Cet exemple crée un instantané de groupe de volumes :

```
apiVersion: groupsnapshot.storage.k8s.io/v1beta1
kind: VolumeGroupSnapshot
metadata:
  name: "vgs1"
  namespace: trident
spec:
  volumeGroupSnapshotClassName: csi-group-snap-class
  source:
    selector:
      matchLabels:
        consistentGroupSnapshot: groupA
```

Récupérer des données de volume à l'aide d'un instantané de groupe

Vous pouvez restaurer des volumes persistants individuels à l'aide des snapshots individuels créés dans le cadre du snapshot de groupe de volumes. Il n'est pas possible de restaurer le snapshot de groupe de volumes en tant qu'unité.

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
```

REMARQUE	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.

Restauration de volumes sur place à partir d'un snapshot

Trident assure une restauration rapide des volumes sur place à partir d'un snapshot à l'aide du `TridentActionSnapshotRestore` système CR (TASR). Cette CR fonctionne comme une action Kubernetes impérative et ne persiste pas une fois l'opération terminée.

Pour plus d'informations, voir "Restauration de volumes sur place à partir d'un snapshot".

Supprimer un PV avec des instantanés de groupe associés

Lors de la suppression d'un instantané de volume de groupe :

- Vous pouvez supprimer VolumeGroupSnapshots dans son intégralité, et non des snapshots individuels du groupe.
- Si des volumes persistants sont supprimés alors qu'un instantané existe pour ce volume persistant, Trident déplacera ce volume vers un état de « suppression » car l'instantané doit être supprimé avant que le volume puisse être supprimé en toute sécurité.
- Si un clone a été créé à l'aide d'un instantané groupé et que le groupe doit ensuite être supprimé, une opération de division sur clone commencera et le groupe ne pourra pas être supprimé tant que la division n'est pas terminée.

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-
8.2/client/config/crd/groupsnapshot.storage.k8s.io_volumegroupsnapshotcl
asses.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
8.2/client/config/crd/groupsnapshot.storage.k8s.io_volumegroupsnapshotco
ntents.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
8.2/client/config/crd/groupsnapshot.storage.k8s.io_volumegroupsnapshots.
yaml
```

2. Créer le contrôleur de snapshot.

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-8.2/deploy/kubernetes/snapshot-
controller/rbac-snapshot-controller.yaml
```

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-8.2/deploy/kubernetes/snapshot-
controller/setup-snapshot-controller.yaml
```

REMARQUE	Si nécessaire, ouvrir `deploy/kubernetes/snapshot-controller/rbac-snapshot-controller.yaml` et mettre à jour `namespace` à votre espace de noms.

Liens connexes

- "Classe d'instantanés de groupe de volumes"
- "Snapshots de volume"

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.
- 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 directement mettre à niveau la version 24.06 (ou toute version 24.06) vers la version 25.06.
- 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 utilisez la version 23.07 et que vous souhaitez effectuer une mise à niveau vers la version 25.06 :
  a. Première mise à jour du 23.07 au 24.06.
  b. Puis passez de 24.06 à 25.06.

REMARQUE	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 règle générale, vous devez effectuer la mise à niveau en utilisant la même méthode que celle utilisée lors de l'installation initiale, cependant vous pouvez "passer d'une méthode d'installation à l'autre". Il existe deux options pour mettre à niveau Trident.

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

ASTUCE	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'un des nombreux "Avantages de l'utilisation de l'opérateur Trident" moyens d'installer et de mettre à niveau Trident est la gestion automatique des objets Trident et Kubernetes sans perturber les volumes montés existants. De cette façon, Trident peut prendre en charge les mises à niveau sans interruption, ou "*mises à jour en continu*". Plus précisément, 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.

ASTUCE	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, soit manuellement, soit à l'aide de Helm. Vous pouvez mettre à niveau une installation de l'opérateur Trident vers une autre installation de l'opérateur Trident ou effectuer une mise à niveau d'une installation `tridentctl` vers une version opérateur Trident. Examinez "Sélectionnez une méthode de mise à niveau" avant de mettre à niveau 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 à l'échelle d'un cluster vers une autre installation d'opérateur Trident à l'échelle d'un cluster. Toutes les versions de Trident utilisent un opérateur à portée de cluster.

REMARQUE	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 1.25 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. Mettre à jour le `operator.yaml` , `tridentorchestrator_cr.yaml` , et `post_1_25_bundle.yaml` avec le registre et les chemins d'image pour la version vers laquelle vous effectuez la mise à niveau (par exemple 25.06) et le secret correct.
3. Supprimez l'opérateur Trident qui a été utilisé pour installer l'instance Trident actuelle. Par exemple, si vous effectuez une mise à niveau à partir de la version 25.02, exécutez la commande suivante :

```
kubectl delete -f 25.02.0/trident-installer/deploy/<bundle.yaml> -n
trident
```

4. 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.
5. Installez Trident en utilisant le fichier YAML approprié pour votre environnement, où *<bundle.yaml>* est `bundle_pre_1_25.yaml` ou `bundle_post_1_25.yaml` en fonction de votre version de Kubernetes. Par exemple, si vous installez Trident 25.06.0, exécutez la commande suivante :

```
kubectl create -f 25.06.0/trident-installer/deploy/<bundle.yaml> -n
trident
```

6. Modifiez le torc trident pour inclure l'image 25.06.0.

Mettre à niveau une installation Helm

Vous pouvez mettre à niveau une installation Trident Helm.

AVERTISSEMENT	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 100.2602.0` pour effectuer la mise à niveau en une seule étape. Si vous n'avez pas ajouté le dépôt Helm ou si vous ne pouvez pas l'utiliser pour effectuer la mise à niveau :
   a. Téléchargez la dernière version de Trident sur "La section *Assets* sur GitHub".
   b. Utilisez la commande `helm upgrade` où `trident-operator-26.06.0.tgz` correspond à la version vers laquelle vous souhaitez effectuer la mise à niveau.

```
helm upgrade <name> trident-operator-26.06.0.tgz
```

REMARQUE	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.

REMARQUE	Avant de changer de méthode d'installation, consultez "Passage d'une méthode d'installation à l'autre".

Étapes

1. Téléchargez la dernière version de Trident.

```
# Download the release required [26.06.0]
mkdir 26.06.0
cd 26.06.0
wget
https://github.com/NetApp/trident/releases/download/v26.06.0/trident-
installer-26.06.0.tar.gz
tar -xf trident-installer-26.06.0.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:                v26.06.0
```

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 "Installez Trident à l'aide de tridentctl".

IMPORTANT	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.

REMARQUE	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.

AVERTISSEMENT	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` : L'adresse/le port d'un proxy pour l'envoi de la télémétrie Autosupport.
`--enable-node-prep` : Tenter d'installer les packages requis sur les nœuds.
`--generate-custom-yaml` : Générez des fichiers YAML sans rien installer.
`-h` , `--help` : Aide à l'installation.
`--http-request-timeout` : Remplacez le délai d'expiration de la requête HTTP pour l'API REST du contrôleur Trident (par défaut 1m30s).
`--image-registry string` : L'adresse/le port d'un registre d'images interne.
`--k8s-timeout duration` : Le délai d'expiration pour toutes les opérations Kubernetes (par défaut 3m0s).
`--kubelet-dir string` : L'emplacement de l'hôte de l'état interne de kubelet (par défaut « /var/lib/kubelet »).
`--log-format string` : Le format de journalisation Trident (texte, json) (par défaut « texte »).
`--node-prep` : Permet à Trident de préparer les nœuds du cluster Kubernetes pour gérer les volumes à l'aide du protocole de stockage de données spécifié. **Actuellement, `iscsi` est la seule valeur prise en charge. À partir d'OpenShift 4.19, la version minimale de Trident prise en charge pour cette fonctionnalité est 25.06.1.**
`--pv string` : Le nom du PV hérité utilisé par Trident, garantit que celui-ci n'existe pas (par défaut « trident »).
`--pvc string` : Le nom du PVC hérité utilisé par Trident, garantit que celui-ci n'existe pas (par défaut « trident »).
`--silence-autosupport` : N'envoyez pas automatiquement les bundles d'autosupport à NetApp (valeur par défaut : true).
`--silent` : Désactivez la plupart des sorties pendant l'installation.
`--trident-image string` : L'image du Trident à installer.
`--k8s-api-qps` :La limite de requêtes par seconde (QPS) pour les requêtes API Kubernetes (par défaut 100 ; facultatif).
`--use-custom-yaml` : Utilisez tous les fichiers YAML existants qui existent dans le répertoire d'installation.
`--use-ipv6` :Utilisez 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. Mettre à jour le backend en utilisant le `tridentctl update backend` commande et le chemin vers la mise à jour `backend.json` déposer.

   Exemple: `tridentctl update backend -f /<path to backend JSON file>/backend.json -n trident`

```
{
  "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.

REMARQUE	Par défaut, les métriques de Trident sont exposées sur le port cible `8001` à `/metrics` point final. Ces indicateurs sont **activés par défaut** lorsque Trident est installé. Vous pouvez configurer la consommation des métriques Trident via HTTPS sur le port `8444` aussi.

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 métriques et obtenir des informations sur les

backends gérés par Trident , les volumes qu'il crée, etc. Voir"Documentation de l'opérateur Prometheus" .

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 ServiceMonitor récupère les métriques renvoyées par le `trident-csi` le service et recherche spécifiquement le `metrics` point de terminaison du service. Par conséquent, Prometheus est désormais configuré pour comprendre les indicateurs 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".

Exploiter les métriques Trident via HTTPS

Pour consommer les métriques Trident via HTTPS (port 8444), vous devez modifier la définition ServiceMonitor pour inclure la configuration TLS. Vous devez également copier le `trident-csi` secret du `trident` espace de noms vers l'espace de noms où Prometheus est exécuté. Vous pouvez le faire en utilisant la commande suivante :

```
kubectl get secret trident-csi -n trident -o yaml | sed 's/namespace:
trident/namespace: monitoring/' | kubectl apply -f -
```

Voici un exemple de ServiceMonitor pour les métriques HTTPS :

```
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:
    - interval: 15s
      path: /metrics
      port: https-metrics
      scheme: https
      tlsConfig:
        ca:
          secret:
            key: caCert
            name: trident-csi
        cert:
          secret:
            key: clientCert
            name: trident-csi
        keySecret:
          key: clientKey
          name: trident-csi
        serverName: trident-csi
```

Trident prend en charge les métriques HTTPS dans toutes les méthodes d'installation : tridentctl, graphique Helm et opérateur :

- Si vous utilisez le `tridentctl install` commande, vous pouvez passer le `--https-metrics` Indicateur permettant d'activer les métriques HTTPS.
- Si vous utilisez le graphique Helm, vous pouvez configurer le `httpsMetrics` Paramètre permettant d'activer les métriques HTTPS.
- Si vous utilisez des fichiers YAML, vous pouvez ajouter le `--https_metrics` drapeau au `trident-main` conteneur dans le `trident-deployment.yaml` déposer.

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

REMARQUE	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".
- Trident AutoSupport ne collecte ni ne transmet d'informations personnelles identifiables (PII) ni d'informations personnelles. Il est fourni avec une "CLUF" qui ne s'applique pas à l'image de conteneur Trident elle-même. Vous pouvez en apprendre davantage sur l'engagement de NetApp en matière de sécurité des données et de confiance "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: null
      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
```

AVERTISSEMENT	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
```

REMARQUE	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
```

REMARQUE 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.

REMARQUE

- 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
```

Outils FC

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

- Lors de l'utilisation de nœuds worker exécutant RHEL/Red Hat Enterprise Linux CoreOS (RHCOS) avec FC PVS, spécifiez la `discard` mountOption dans la classe de stockage pour effectuer la récupération d'espace en ligne. Reportez-vous à la "Documentation Red Hat".

RHEL 8+

1. Installez les packages système suivants :

```
sudo yum install -y lsscsi device-mapper-multipath
```

2. Activer les chemins d'accès multiples :

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

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

3. Assurez-vous que `multipathd` est en cours d'exécution :

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

Ubuntu

1. Installez les packages système suivants :

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

2. 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
```

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

3. Assurez-vous que `multipath-tools` est activé et en cours d'exécution :

```
sudo systemctl status multipath-tools
```

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)

REMARQUE	**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/10.0/trident-
   installer-26.02.0.tar.gz
   tar zxf trident-installer-26.02.0.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
```

REMARQUE 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
   ```

REMARQUE	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
```

REMARQUE	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
```

IMPORTANT

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
```

REMARQUE	Si vous n'en spécifiez aucun "options", les valeurs par défaut du pilote sont utilisées.

- Remplacez la taille de volume par défaut. Consultez l'exemple suivant pour créer un volume de 20 Gio avec un pilote :

```
docker volume create -d netapp --name my_vol --opt size=20G
```

ASTUCE	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
```

IMPORTANT	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

Lors de l'utilisation du `ontap-nas` , `ontap-san` , et `solidfire-san` Les pilotes de stockage Trident peuvent cloner des volumes. Lors de l'utilisation du `ontap-nas-flexgroup` ou `ontap-nas-economy` Le clonage des pilotes n'est pas pris en charge. La création d'un nouveau volume à partir d'un volume existant entraînera la création d'un nouvel instantané.

- 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, iSCSI et FC 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`, ce qui signifie qu'aucun instantané ne sera automatiquement créé pour le volume. Sauf modification par votre administrateur du stockage, il existe une règle nommée « par défaut » sur tous les systèmes ONTAP qui crée et conserve six copies Snapshot par heure, deux par jour et deux par semaine. Les données conservées dans un snapshot peuvent être restaurées en naviguant vers le `.snapshot` répertoire de n'importe quel répertoire du volume.

Option	Description
`snapshotReserve`	La réserve d'instantanés sera alors définie sur le pourcentage souhaité. La valeur par défaut n'est pas définie. Cela signifie que ONTAP sélectionne la fonction de copie instantanée (généralement 5 %) si vous avez sélectionné une stratégie de snapshots, ou 0 % si la stratégie de snapshots n'est pas définie. Vous pouvez définir la valeur par défaut des snapshots dans le fichier de configuration pour tous les systèmes back-end ONTAP. Vous pouvez l'utiliser comme option de création de volumes pour tous les systèmes back-end ONTAP, à l'exception des économies ontap-nas.
`splitOnClone`	Lors du clonage d'un volume, ONTAP va immédiatement séparer le clone de son volume parent. La valeur par défaut est `false`. Pour optimiser l'efficacité du stockage, il est préférable de séparer le clone de son parent dès sa création, car il est peu probable que cette utilisation soit utile. Par exemple, le clonage d'une base de données vide permet de gagner beaucoup de temps mais réduit les économies de stockage. Il est donc préférable de séparer immédiatement le clone.
`encryption`	Activez NetApp Volume Encryption (NVE) sur le nouveau volume. La valeur par défaut est `false`. Pour utiliser cette option, NVE doit être sous licence et activé sur le cluster. Si NAE est activé sur le back-end, 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.

Option	Description
`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`.
`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 :

- Créer un volume de 10 Gio :

```
docker volume create -d netapp --name demo -o size=10G -o
encryption=true
```

- Créer un volume de 100 Gio avec des instantanés :

```
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 20 Mio.

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 de la 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 1 Gio ou entrée de configuration… "defaults": {"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.

- Créer un volume Gold de 10 Gio :

```
docker volume create -d solidfire --name sfGold -o type=Gold -o size=10G
```

- Créer 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_`
`limitVolumeSize`	Restriction facultative sur les tailles de volume. Par défaut : « » (non appliqué)	`10g`

ASTUCE

N'utilisez pas `storagePrefix` (y compris la valeur par défaut) les systèmes back-end Element. Par défaut, le `solidfire-san` pilote ignore ce paramètre et n'utilise pas de préfixe. NetApp recommande d'utiliser soit un tenantID spécifique pour le mappage de volume Docker, soit des données d'attribut renseignées sur la version Docker, les informations sur le pilote et le nom brut de Docker au cas où un nom munging aurait pu être utilisé.

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**: NetApp recommande 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, ce qui vous permet de créer un DNS circulaire pour équilibrer la charge sur plusieurs dataLIFs. **Pilotes SAN ONTAP** : ne spécifiez pas pour iSCSI ou FC. Trident utilise "Mappage de LUN sélectif ONTAP" pour détecter les LIF iSCSI ou FC requises pour établir 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 précis des options de montage NFS ; valeur par défaut : « -o nfsvers=3 ». **Disponible uniquement pour les `ontap-nas` pilotes et `ontap-nas-economy`.** "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).	`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.	Active ou désactive l'accès au `.snapshot` répertoire. Valeurs valides : `true`, `false`. Le comportement par défaut peut varier selon la version d'ONTAP et de Trident. Définissez explicitement pour garantir un comportement cohérent.
`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`

Option	Description	Exemple
`tieringPolicy`	Règle de hiérarchisation à utiliser, la valeur par défaut est `none`.	`none`
`skipRecoveryQueue`	Lors de la suppression d'un volume, ignorez la file d'attente de récupération dans le stockage et supprimez le volume immédiatement.	``

Options d'évolutivité

Les `ontap-nas` pilotes et `ontap-san` créent un ONTAP FlexVol pour chaque volume Docker. ONTAP prend en charge jusqu'à 1000 volumes FlexVol par nœud de cluster avec un maximum de 12,000 volumes FlexVol si vos exigences de volume Docker correspondent à cette limitation, le `ontap-nas` pilote est la solution NAS préférée en raison des fonctionnalités supplémentaires proposées par FlexVols, telles que les copies Snapshot granulaires de volume Docker et le clonage.

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` pilote crée des volumes Docker sous forme de qtrees ONTAP au sein d'un pool de volumes FlexVol automatiquement gérés. 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` pilote ne prend pas en charge les snapshots granulaires au niveau des volumes Docker ni le clonage.

REMARQUE	Le `ontap-nas-economy` pilote n'est pas pris en charge dans Docker Swarm, car Docker Swarm n'orchestre pas la création de volumes sur plusieurs nœuds.

Le `ontap-san-economy` pilote crée des volumes Docker en tant que LUN ONTAP au sein d'un pool partagé de volumes FlexVol automatiquement gérés. 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 obtenir des informations sur les FlexGroups et les charges de travail adaptées aux FlexGroups, reportez-vous à la "Guide des meilleures pratiques et d'implémentation des volumes NetApp FlexGroup".

Pour bénéficier de fonctionnalités avancées et d'une grande échelle dans un même environnement, vous pouvez exécuter plusieurs instances du plug-in de volume Docker, l'une utilisant et l' `ontap-nas`autre avec `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"
security login create -username <user_name\> -application http -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
}
```

Exemple SCSI sur FC pour le pilote <code> ONTAP-</code>

```
{
  "version": 1,
  "backendName": "ontap-san-backend",
  "storageDriverName": "ontap-san",
  "managementLIF": "10.0.0.1",
  "sanType": "fcp",
  "svm": "trident_svm",
  "username": "vsadmin",
  "password": "password",
  "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`

Option	Description	Exemple
`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.

REMARQUE	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 pilotes sont soumis 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 « gagnant » clair ; par exemple, Element dispose d'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 FlexVol. Avoir le volume maximum réglé sur 1250 semble très raisonnable. Toutefois, si seul un nœud est attribué au SVM ou si "64 bits" 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 la cible de tous les volumes provisionnés Trident. Cela signifie que la limite de volume peut être atteinte pour ce nœud avant que la valeur ne `max-volumes` soit atteinte, ce qui a un impact sur les opérations de volume Trident et autres qui utilisent 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.**

Clonez un volume

NetApp Trident prend en charge le clonage de volumes lors de l'utilisation de `ontap-nas` , `ontap-san` , et `solidfire-san` pilotes de stockage. Lors de l'utilisation du `ontap-nas-flexgroup` ou `ontap-nas-economy` Le clonage des pilotes n'est pas pris en charge. La création d'un nouveau volume à partir d'un volume existant entraînera la création d'un nouvel instantané.

AVERTISSEMENT	Évitez de cloner un PVC associé à une autre StorageClass. Effectuez des opérations de clonage au sein de la même StorageClass pour garantir la compatibilité et éviter tout comportement inattendu.

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.

IMPORTANT	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 QoS, reportez-vous "Référence des commandes ONTAP" à la section .

Limitez l'accès aux ressources de stockage aux membres du cluster Kubernetes

La limitation de l'accès aux volumes NFS, aux LUN iSCSI et aux LUN FC créés par Trident est un composant essentiel de la stratégie 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

La `showmount` fonctionnalité permet à un client NFS d'interroger le SVM sur la liste des exports NFS disponibles. Un pod déployé dans le cluster Kubernetes peut exécuter la `showmount -e` commande sur et recevoir la liste des montages disponibles, y compris ceux auxquels il 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

REMARQUE 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 des blocs et la bande passante influent directement sur le nombre d'IOPS. Lorsque la taille des blocs augmente, le système accroît la bande passante jusqu'au niveau nécessaire pour traiter les blocs plus volumineux. Lorsque la bande passante augmente, le nombre d'IOPS que le système peut atteindre diminue. Consultez "Qualité de service SolidFire" pour plus d'informations sur la QoS 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.

REMARQUE 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"
- "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`	Note de bas de page : 3[]	Note de bas de page : 3[]	Yes [5]	Oui.	Note de bas de page : 3[]	Oui.	Note de bas de page : 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]
`ontap-san-economy`	Oui.	Oui.	Yes [4]	Oui.	Note de bas de page : 3[]	Oui.	Note de bas de page : 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 NO [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.

REMARQUE	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` driver 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

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 backend avec les mêmes identifiants de stockage et un ensemble de paramètres différent. L'introduction des pools virtuels a résolu ce problème. Un pool virtuel est une abstraction de niveau introduite entre le backend et la classe de stockage Kubernetes. L'administrateur peut ainsi définir des paramètres ainsi que des étiquettes référencées via les classes de stockage Kubernetes comme sélecteur, indépendamment du backend. Les pools virtuels peuvent être définis pour tous les backends NetApp pris en charge par Trident. Cette liste comprend SolidFire/ NetApp HCI, ONTAP, ainsi Azure NetApp Files.

REMARQUE	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.

REMARQUE	Les provisionneurs in-Tree Kubernetes ne prennent pas en charge les opérations de redimensionnement des volumes pour NFS, iSCSI ou FC PVS pour le moment. Trident prend en charge l'extension des volumes NFS, iSCSI et FC.

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, iSCSI et FC. Cela permet aux utilisateurs de 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 et les backends HCI SolidFire/ NetApp HCI . Pour permettre une éventuelle extension ultérieure, définissez `allowVolumeExpansion` à `true` dans votre StorageClass associée au volume. Lorsque le volume persistant doit être redimensionné, modifiez le `spec.resources.requests.storage` annotation dans la revendication de volume persistant à la taille de volume requise. Trident se chargera 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. Ceci est actuellement pris en charge par le `ontap-nas` , `ontap-nas-flexgroup` , `solidfire-san` , et `azure-netapp-files` conducteurs. Cette fonctionnalité est utile lors du portage d'une application existante vers Kubernetes ou lors de scénarios de reprise après sinistre.

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` Le pilote est utilisé, utilisez la commande `tridentctl import volume <backend-name> <volume path> -f /path/pvc.yaml` pour importer le volume dans Kubernetes afin qu'il soit géré 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.

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.

AVERTISSEMENT	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.

REMARQUE	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 "Documentation de connexion Red Hat OpenShift"qui doivent être vérifiées, configurées et utilisées 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 dataLIF de 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 varier légèrement avec les différentes versions d'OpenShift. Assurez-vous de lire et de suivre "Documentation de Red Hat OpenShift Container Platform 3.11"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.

REMARQUE	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 dataLIF de 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. Assurez-vous de vérifier et de suivre "Guide de déploiement de Red Hat OpenShift"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 volume

REMARQUE	NetApp recommande de protéger la SVM où réside la FlexVol avec une relation SnapMirror avec 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.

REMARQUE	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".

REMARQUE	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"

Automatisation du basculement des applications avec état grâce à Trident

La fonction de détachement forcé de Trident vous permet de détacher automatiquement les volumes des nœuds défaillants d'un cluster Kubernetes, évitant ainsi la corruption des données et garantissant la disponibilité des applications. Cette fonctionnalité est particulièrement utile dans les scénarios où les nœuds ne répondent plus ou sont mis hors ligne pour maintenance.

Détails sur le détachement forcé

Le détachement forcé est disponible pour `ontap-san` , `ontap-san-economy` , `ontap-nas` , et `ontap-`

`nas-economy` seulement. Avant d'activer le détachement forcé, l'arrêt non gracieux du nœud (NGNS) doit être activé sur le cluster Kubernetes. NGNS est activé par défaut pour Kubernetes 1.28 et supérieur. Pour plus d'informations, reportez-vous à "Kubernetes : arrêt du nœud sans interruption" .

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

AVERTISSEMENT	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. Interrompre l'accès E/S du backend pour les volumes montés sur ce nœud.
2. Marquer l'objet de nœud Trident comme `dirty` (non sécurisé pour les nouvelles publications).

REMARQUE	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.

Détails sur le basculement automatique

Vous pouvez automatiser le processus de détachement forcé grâce à l'intégration avec"opérateur de vérification de l'état des nœuds (NHC)" . Lorsqu'une panne de nœud survient, NHC déclenche la remédiation du nœud Trident (TNR) et force le détachement automatiquement en créant une CR TridentNodeRemediation dans l'espace de noms Trident définissant le nœud défaillant. Un TNR est créé uniquement en cas de défaillance d'un nœud, et supprimé par le NHC une fois que le nœud est remis en ligne ou qu'il est supprimé.

Échec du processus de suppression du pod du nœud

Le basculement automatique sélectionne les charges de travail à retirer du nœud défaillant. Lorsqu'un TNR est créé, le contrôleur TNR marque le nœud comme étant sale, empêchant toute nouvelle publication de volume et commence à supprimer les pods pris en charge par détachement forcé et leurs attachements de volume.

Tous les volumes/PVC pris en charge par le détachement forcé sont pris en charge par le basculement automatique :

- Volumes NAS et NAS-economy utilisant des politiques d'exportation automatique (SMB n'est pas encore pris en charge).
- Volumes SAN et SAN-économie.

Se référer àDétails sur le détachement forcé .

Comportement par défaut :

- Les pods utilisant des volumes pris en charge par le détachement forcé sont supprimés du nœud défaillant. Kubernetes va les reprogrammer sur un nœud sain.
- Les pods utilisant un volume non pris en charge par le détachement forcé, y compris les volumes non-Trident, ne sont pas supprimés du nœud défaillant.
- Les pods sans état (et non les PVC) ne sont pas supprimés du nœud défaillant, sauf si l'annotation du pod est présente. `trident.netapp.io/podRemediationPolicy: delete` est réglé.

Remplacement du comportement de suppression des pods :

Le comportement de suppression des pods peut être personnalisé à l'aide d'une annotation de pod : `trident.netapp.io/podRemediationPolicy[retain, delete]` . Ces annotations sont examinées et utilisées en cas de basculement. Appliquez des annotations à la spécification du pod de déploiement/replicaset Kubernetes pour empêcher leur disparition après un basculement :

- `retain`- Le pod ne sera PAS supprimé du nœud défaillant lors d'un basculement automatique.
- `delete`- Le pod sera supprimé du nœud défaillant lors d'un basculement automatique.

Ces annotations peuvent être appliquées à n'importe quel pod.

AVERTISSEMENT	• Les opérations d'E/S seront bloquées uniquement sur les nœuds défaillants pour les volumes prenant en charge le détachement forcé. • Pour les volumes qui ne prennent pas en charge le détachement forcé, il existe un risque de corruption des données et de problèmes de connexion multiple.

Remédiation des nœuds Trident CR

Le CR TridentNodeRemediation (TNR) définit un nœud défaillant. Le nom du TNR correspond au nom du nœud défaillant.

Exemple de fichier TNR :

```
apiVersion: trident.netapp.io/v1
kind: TridentNodeRemediation
metadata:
  name: <K8s-node-name>
spec: {}
```

États TNR : Utilisez les commandes suivantes pour consulter l'état des animaux TNR :
`kubectl get tnr <name> -n <trident-namespace>`

Les animaux TNR peuvent se trouver dans l'un des états suivants :

- *Remédiation* :
  - Cessez l'accès aux E/S du backend pour les volumes pris en charge par détachement forcé montés sur ce nœud.
  - L'objet nœud Trident est marqué comme non conforme (non sûr pour les nouvelles publications).
  - Supprimez les pods et les volumes attachés du nœud
- *Récupération du nœud en attente*:
  - Le contrôleur attend que le nœud soit de nouveau en ligne.
  - Une fois le nœud en ligne, le contrôle de publication garantira que le nœud est propre et prêt pour les nouvelles publications de volumes.
- Si le nœud est supprimé de K8s, le contrôleur TNR supprimera le TNR et cessera la réconciliation.
- *Réussi*:
  - Toutes les étapes de correction et de récupération des nœuds ont été réalisées avec succès. Le nœud est propre et prêt pour la publication de nouveaux volumes.
- *Échoué*:
  - Erreur irrécupérable. Les raisons des erreurs sont définies dans le champ status.message de la CR.

Activation du basculement automatique

Prérequis :

- Assurez-vous que le détachement forcé est activé avant d'activer le basculement automatique. Pour plus d'informations, veuillez consulterDétails sur le détachement forcé .
- Installez le contrôle d'intégrité des nœuds (NHC) dans le cluster Kubernetes.
  - "Installer le SDK de l'opérateur".
  - Installez Operator Lifecycle Manager (OLM) dans le cluster s'il n'est pas déjà installé : `operator-sdk olm install` .
  - Installer l'opérateur de vérification de l'état du nœud : `kubectl create -f https://operatorhub.io/install/node-healthcheck-operator.yaml` .

REMARQUE	Vous pouvez également utiliser d'autres méthodes pour détecter les défaillances de nœud, comme spécifié dans la section Intégration de solutions personnalisées de vérification de l'état des nœuds ci-dessous.

Voir"Opérateur de vérification de l'état des nœuds" pour plus d'informations.

Étapes

1. Créez une CR NodeHealthCheck (NHC) dans l'espace de noms Trident pour surveiller les nœuds de travail du cluster. Exemple:

```
apiVersion: remediation.medik8s.io/v1alpha1
kind: NodeHealthCheck
metadata:
  name: <CR name>
spec:
  selector:
    matchExpressions:
      - key: node-role.kubernetes.io/control-plane
        operator: DoesNotExist
      - key: node-role.kubernetes.io/master
        operator: DoesNotExist
  remediationTemplate:
    apiVersion: trident.netapp.io/v1
    kind: TridentNodeRemediationTemplate
    namespace: <Trident installation namespace>
    name: trident-node-remediation-template
  minHealthy: 0 # Trigger force-detach upon one or more node failures
  unhealthyConditions:
    - type: Ready
      status: "False"
      duration: 0s
    - type: Ready
      status: Unknown
      duration: 0s
```

2. Appliquez la demande de changement (CR) de vérification de l'état du nœud dans le `trident` espace de noms.

```
kubectl apply -f <nhc-cr-file>.yaml -n <trident-namespace>
```

La ressource personnalisée ci-dessus est configurée pour surveiller les nœuds de travail K8s afin de détecter les conditions de nœud Ready : false et Unknown. Le basculement automatique sera déclenché lorsqu'un nœud passera à l'état Prêt : faux ou Prêt : inconnu.

Le `unhealthyConditions` Le CR utilise un délai de grâce de 0 seconde. Cela provoque le déclenchement immédiat du basculement automatique dès que K8s définit la condition du nœud sur Ready: false, ce qui se produit lorsque K8s perd le signal de présence d'un nœud. K8s a un délai par défaut de 40 secondes après le dernier battement de cœur avant de définir Ready sur false. Cette période de grâce peut être personnalisée dans les options de déploiement de K8s.

Pour plus d'options de configuration, consultez"Documentation de l'opérateur Node-Healthcheck" .

Informations de configuration supplémentaires

Lorsque Trident est installé avec le détachement forcé activé, deux ressources supplémentaires sont automatiquement créées dans l'espace de noms Trident pour faciliter l'intégration avec NHC : TridentNodeRemediationTemplate (TNRT) et ClusterRole.

Modèle de remédiation des nœuds Trident (TNRT) :

Le TNRT sert de modèle au contrôleur du NHC, qui utilise le TNRT pour générer des ressources TNR selon les besoins.

```
apiVersion: trident.netapp.io/v1
kind: TridentNodeRemediationTemplate
metadata:
  name: trident-node-remediation-template
  namespace: trident
spec:
  template:
    spec: {}
```

Rôle du cluster :

Un rôle de cluster est également ajouté lors de l'installation lorsque le détachement forcé est activé. Cela confère au NHC les autorisations nécessaires pour les TNR dans l'espace de noms Trident .

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  labels:
    rbac.ext-remediation/aggregate-to-ext-remediation: "true"
  name: tridentnoderemediation-access
rules:
- apiGroups:
  - trident.netapp.io
  resources:
  - tridentnoderemediationtemplates
  - tridentnoderemediations
  verbs:
  - get
  - list
  - watch
  - create
  - update
  - patch
  - delete
```

Mises à niveau et maintenance des clusters K8s

Pour éviter tout basculement, suspendez le basculement automatique pendant les opérations de maintenance ou de mise à niveau de K8s, lorsque les nœuds sont susceptibles de tomber en panne ou de redémarrer. Vous pouvez mettre en pause le CR du NHC (décrit ci-dessus) en modifiant son CR :

```
kubectl patch NodeHealthCheck <cr-name> --patch
'{"spec":{"pauseRequests":["<description-for-reason-of-pause>"]}}' --type=merge
```

Cela interrompt le basculement automatique. Pour réactiver le basculement automatique, supprimez les requêtes de pause de la spécification une fois la maintenance terminée.

Limites

- Les opérations d'E/S sont empêchées uniquement sur les nœuds défaillants pour les volumes pris en charge par le détachement forcé. Seuls les pods utilisant des volumes/PVC pris en charge par le détachement forcé sont automatiquement supprimés.
- Le basculement automatique et le détachement forcé s'exécutent à l'intérieur du module de contrôle Trident. Si le nœud hébergeant le contrôleur Trident tombe en panne, le basculement automatique sera retardé jusqu'à ce que Kubernetes déplace le pod vers un nœud sain.

Intégration de solutions personnalisées de vérification de l'état des nœuds

Vous pouvez remplacer l'opérateur de vérification de l'état du nœud par d'autres outils de détection des pannes de nœuds pour déclencher un basculement automatique. Pour garantir la compatibilité avec le mécanisme de basculement automatique, votre solution personnalisée doit :

- Créez un TNR lorsqu'une défaillance de nœud est détectée, en utilisant le nom du nœud défaillant comme nom de la ressource de changement (CR) du TNR.
- Supprimez le TNR lorsque le nœud a récupéré et que le TNR est à l'état « Réussi ».

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.

REMARQUE	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.
  - Vous pouvez définir l'indicateur de chiffrement NVE `""` pour créer des volumes NAE.
- Si NAE n'est pas activé sur le back-end, tout volume provisionné dans Trident est activé sur NVE, à moins que l'indicateur de chiffrement NVE ne soit défini sur `false` (valeur par défaut) dans la configuration back-end.

REMARQUE	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".

REMARQUE	Le chiffrement LUKS n'est pas pris en charge par les systèmes ASA r2. Pour plus d'informations sur les systèmes ASA r2, consultez la section "En savoir plus sur les systèmes de stockage ASA r2" .

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, NetApp recommande 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.

AVERTISSEMENT	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.

REMARQUE	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.

Chiffrement Kerberos à la volée

Avec le chiffrement à la volée Kerberos, vous pouvez améliorer la sécurité d'accès aux données en activant le chiffrement pour le trafic entre votre cluster géré et le back-end de stockage.

Trident prend en charge le chiffrement Kerberos pour ONTAP en tant que back-end de stockage :

- **On-premise ONTAP** - Trident prend en charge le chiffrement Kerberos sur les connexions NFSv3 et NFSv4 depuis Red Hat OpenShift et les clusters Kubernetes en amont vers les volumes ONTAP sur site.

Vous pouvez créer, supprimer, redimensionner, snapshot, cloner clone en lecture seule et importation des volumes qui utilisent le chiffrement NFS.

Configurez le chiffrement Kerberos à la volée avec les volumes ONTAP sur site

Vous pouvez activer le chiffrement Kerberos sur le trafic de stockage entre votre cluster géré et un système back-end de stockage ONTAP sur site.

REMARQUE	Le chiffrement Kerberos pour le trafic NFS avec les systèmes back-end de stockage ONTAP sur site est uniquement pris en charge à l'aide du `ontap-nas` pilote de stockage.

Avant de commencer

- Vérifiez que vous avez accès à l' `tridentctl` utilitaire.
- Assurez-vous de disposer d'un accès administrateur au système back-end de stockage ONTAP.
- Assurez-vous de connaître le nom du ou des volumes que vous allez partager à partir du back-end de stockage ONTAP.
- Assurez-vous d'avoir préparé la machine virtuelle de stockage ONTAP à prendre en charge le chiffrement Kerberos pour les volumes NFS. Reportez-vous "Activez Kerberos sur une dataLIF" à pour obtenir des instructions.
- Assurez-vous que tous les volumes NFSv4 que vous utilisez avec le chiffrement Kerberos sont correctement configurés. Reportez-vous à la section « Configuration du domaine NFSv4 » (page 13) de NetApp "NetApp NFSv4 : améliorations et guide des bonnes pratiques".

Ajoutez ou modifiez les règles d'export ONTAP

Vous devez ajouter des règles aux règles d'export ONTAP existantes ou créer de nouvelles règles d'export qui prennent en charge le chiffrement Kerberos pour le volume racine de la VM de stockage ONTAP ainsi que tous les volumes ONTAP partagés avec le cluster Kubernetes en amont. Les règles d'export-policy que vous ajoutez ou les nouvelles règles d'export que vous créez doivent prendre en charge les protocoles d'accès et autorisations d'accès suivants :

Protocoles d'accès

Configurez la règle d'export avec les protocoles d'accès NFS, NFSv3 et NFSv4.

Accédez aux informations

Vous pouvez configurer l'une des trois versions différentes du cryptage Kerberos, en fonction de vos besoins pour le volume :

- **Kerberos 5** - (authentification et cryptage)
- **Kerberos 5i** - (authentification et chiffrement avec protection d'identité)
- **Kerberos 5p** - (authentification et chiffrement avec protection de l'identité et de la vie privée)

Configurez la règle d'export ONTAP avec les autorisations d'accès appropriées. Par exemple, si les clusters montant les volumes NFS avec un mélange de cryptage Kerberos 5i et Kerberos 5p, utilisez les paramètres d'accès suivants :

Type	Accès en lecture seule	Accès en lecture/écriture	Accès superutilisateur
UNIX	Activé	Activé	Activé
Kerberos 5i	Activé	Activé	Activé
Kerberos 5p	Activé	Activé	Activé

Pour plus d'informations sur la création de règles d'export ONTAP et de règles d'export-policy, reportez-vous à la documentation suivante :

- "Créer une export-policy"
- "Ajouter une règle à une export-policy"

Créer un back-end de stockage

Vous pouvez créer une configuration back-end de stockage Trident qui inclut une fonctionnalité de chiffrement Kerberos.

Description de la tâche

Lorsque vous créez un fichier de configuration du back-end de stockage qui configure le chiffrement Kerberos, vous pouvez spécifier l'une des trois versions différentes du chiffrement Kerberos à l'aide du `spec.nfsMountOptions` paramètre :

- `spec.nfsMountOptions: sec=krb5` (authentification et chiffrement)
- `spec.nfsMountOptions: sec=krb5i` (authentification et chiffrement avec protection de l'identité)
- `spec.nfsMountOptions: sec=krb5p` (authentification et chiffrement avec protection de l'identité et de la confidentialité)

Spécifiez un seul niveau Kerberos. Si vous spécifiez plusieurs niveaux de cryptage Kerberos dans la liste des paramètres, seule la première option est utilisée.

Étapes

1. Sur le cluster géré, créez un fichier de configuration du back-end de stockage à l'aide de l'exemple suivant. Remplacez les valeurs entre parenthèses <> par les informations de votre environnement :

```
apiVersion: v1
kind: Secret
metadata:
  name: backend-ontap-nas-secret
type: Opaque
stringData:
  clientID: <CLIENT_ID>
  clientSecret: <CLIENT_SECRET>
---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-ontap-nas
spec:
  version: 1
  storageDriverName: "ontap-nas"
  managementLIF: <STORAGE_VM_MGMT_LIF_IP_ADDRESS>
  dataLIF: <PROTOCOL_LIF_FQDN_OR_IP_ADDRESS>
  svm: <STORAGE_VM_NAME>
  username: <STORAGE_VM_USERNAME_CREDENTIAL>
  password: <STORAGE_VM_PASSWORD_CREDENTIAL>
  nasType: nfs
  nfsMountOptions: ["sec=krb5i"] #can be krb5, krb5i, or krb5p
  qtreesPerFlexvol:
  credentials:
    name: backend-ontap-nas-secret
```

2. Utilisez le fichier de configuration que vous avez créé à l'étape précédente pour créer le backend :

```
tridentctl create backend -f <backend-configuration-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.

Créer une classe de stockage

Vous pouvez créer une classe de stockage pour provisionner des volumes avec le chiffrement Kerberos.

Description de la tâche

Lorsque vous créez un objet classe de stockage, vous pouvez spécifier l'une des trois versions différentes du chiffrement Kerberos à l'aide du `mountOptions` paramètre :

- `mountOptions: sec=krb5` (authentification et chiffrement)
- `mountOptions: sec=krb5i` (authentification et chiffrement avec protection de l'identité)
- `mountOptions: sec=krb5p` (authentification et chiffrement avec protection de l'identité et de la confidentialité)

Spécifiez un seul niveau Kerberos. Si vous spécifiez plusieurs niveaux de cryptage Kerberos dans la liste des paramètres, seule la première option est utilisée. Si le niveau de chiffrement que vous avez spécifié dans la configuration du back-end de stockage est différent du niveau que vous spécifiez dans l'objet classe de stockage, l'objet classe de stockage a priorité.

Étapes

1. Créez un objet StorageClass Kubernetes à l'aide de l'exemple suivant :

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-nas-sc
provisioner: csi.trident.netapp.io
mountOptions:
  - sec=krb5i #can be krb5, krb5i, or krb5p
parameters:
  backendType: ontap-nas
  storagePools: ontapnas_pool
  trident.netapp.io/nasType: nfs
allowVolumeExpansion: true
```

2. Créer la classe de stockage :

```
kubectl create -f sample-input/storage-class-ontap-nas-sc.yaml
```

3. Assurez-vous que la classe de stockage a été créée :

```
kubectl get sc ontap-nas-sc
```

Vous devez voir les résultats similaires à ce qui suit :

```
NAME          PROVISIONER             AGE
ontap-nas-sc    csi.trident.netapp.io   15h
```

Provisionner les volumes

Une fois que vous avez créé un système back-end et une classe de stockage, vous pouvez provisionner un volume. Pour obtenir des instructions, reportez-vous à "Provisionner un volume" la .

Configurez le chiffrement Kerberos à la volée avec les volumes Azure NetApp Files

Vous pouvez activer le chiffrement Kerberos sur le trafic de stockage entre votre cluster géré et un seul système back-end de stockage Azure NetApp Files ou un pool virtuel de systèmes back-end de stockage Azure NetApp Files.

Avant de commencer

- Vérifiez que vous avez activé Trident sur le cluster Red Hat OpenShift géré.
- Vérifiez que vous avez accès à l' `tridentctl` utilitaire.
- Assurez-vous d'avoir préparé le back-end de stockage Azure NetApp Files pour le chiffrement Kerberos en notant les exigences et en suivant les instructions de la section "Documentation Azure NetApp Files".
- Assurez-vous que tous les volumes NFSv4 que vous utilisez avec le chiffrement Kerberos sont correctement configurés. Reportez-vous à la section « Configuration du domaine NFSv4 » (page 13) de NetApp "NetApp NFSv4 : améliorations et guide des bonnes pratiques".

Créer un back-end de stockage

Vous pouvez créer une configuration back-end de stockage Azure NetApp Files qui inclut une fonctionnalité de chiffrement Kerberos.

Description de la tâche

Lorsque vous créez un fichier de configuration du back-end de stockage qui configure le cryptage Kerberos, vous pouvez le définir de manière à ce qu'il soit appliqué à l'un des deux niveaux possibles :

- Le **niveau du backend de stockage** utilisant le `spec.kerberos` champ
- **Niveau de pool virtuel** utilisant le `spec.storage.kerberos` champ

Lorsque vous définissez la configuration au niveau du pool virtuel, le pool est sélectionné à l'aide du libellé de la classe de stockage.

À chaque niveau, vous pouvez spécifier l'une des trois versions différentes du cryptage Kerberos :

- `kerberos: sec=krb5` (authentification et chiffrement)
- `kerberos: sec=krb5i` (authentification et chiffrement avec protection de l'identité)
- `kerberos: sec=krb5p` (authentification et chiffrement avec protection de l'identité et de la confidentialité)

Étapes

1. Sur le cluster géré, créez un fichier de configuration back-end de stockage en utilisant l'un des exemples suivants, selon l'endroit où vous devez définir le back-end de stockage (niveau du back-end de stockage ou niveau du pool virtuel). Remplacez les valeurs entre parenthèses <> par les informations de votre environnement :

Exemple au niveau du back-end de stockage

```
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-secret
type: Opaque
stringData:
  clientID: <CLIENT_ID>
  clientSecret: <CLIENT_SECRET>

---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc
spec:
  version: 1
  storageDriverName: azure-netapp-files
  subscriptionID: <SUBSCRIPTION_ID>
  tenantID: <TENANT_ID>
  location: <AZURE_REGION_LOCATION>
  serviceLevel: Standard
  networkFeatures: Standard
  capacityPools: <CAPACITY_POOL>
  resourceGroups: <RESOURCE_GROUP>
  netappAccounts: <NETAPP_ACCOUNT>
  virtualNetwork: <VIRTUAL_NETWORK>
  subnet: <SUBNET>
  nasType: nfs
  kerberos: sec=krb5i #can be krb5, krb5i, or krb5p
  credentials:
    name: backend-tbc-secret
```

Exemple de pool virtuel

```
---
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-secret
type: Opaque
stringData:
  clientID: <CLIENT_ID>
  clientSecret: <CLIENT_SECRET>

---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc
spec:
  version: 1
  storageDriverName: azure-netapp-files
  subscriptionID: <SUBSCRIPTION_ID>
  tenantID: <TENANT_ID>
  location: <AZURE_REGION_LOCATION>
  serviceLevel: Standard
  networkFeatures: Standard
  capacityPools: <CAPACITY_POOL>
  resourceGroups: <RESOURCE_GROUP>
  netappAccounts: <NETAPP_ACCOUNT>
  virtualNetwork: <VIRTUAL_NETWORK>
  subnet: <SUBNET>
  nasType: nfs
  storage:
    - labels:
        type: encryption
      kerberos: sec=krb5i #can be krb5, krb5i, or krb5p
  credentials:
    name: backend-tbc-secret
```

2. Utilisez le fichier de configuration que vous avez créé à l'étape précédente pour créer le backend :

```
tridentctl create backend -f <backend-configuration-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.

Créer une classe de stockage

Vous pouvez créer une classe de stockage pour provisionner des volumes avec le chiffrement Kerberos.

Étapes

1. Créez un objet StorageClass Kubernetes à l'aide de l'exemple suivant :

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: sc-nfs
provisioner: csi.trident.netapp.io
parameters:
  backendType: azure-netapp-files
  trident.netapp.io/nasType: nfs
  selector: type=encryption
```

2. Créer la classe de stockage :

```
kubectl create -f sample-input/storage-class-sc-nfs.yaml
```

3. Assurez-vous que la classe de stockage a été créée :

```
kubectl get sc -sc-nfs
```

Vous devez voir les résultats similaires à ce qui suit :

```
NAME          PROVISIONER             AGE
sc-nfs        csi.trident.netapp.io   15h
```

Provisionner les volumes

Une fois que vous avez créé un système back-end et une classe de stockage, vous pouvez provisionner un volume. Pour obtenir des instructions, reportez-vous à "Provisionner un volume" la .

Protégez vos applications avec Trident Protect

Découvrez Trident Protect

NetApp Trident Protect offre des fonctionnalités avancées de gestion des données d'application qui améliorent la fonctionnalité et la disponibilité des applications Kubernetes avec état, prises en charge par les systèmes de stockage NetApp ONTAP et le provisionneur de stockage NetApp Trident CSI. Trident Protect simplifie la gestion, la protection et le déplacement des charges de travail conteneurisées entre les clouds publics et les environnements sur site. Il offre également des capacités d'automatisation via son API et sa CLI.

Vous pouvez protéger les applications avec Trident Protect en créant des ressources personnalisées (CR) ou en utilisant l'interface de ligne de commande (CLI) de Trident Protect.

Et la suite ?

Vous pouvez vous renseigner sur les exigences de Trident Protect avant de l'installer :

- "Exigences de Trident Protect"

Installer Trident Protect

Exigences de Trident Protect

Commencez par vérifier l'état de préparation de votre environnement opérationnel, de vos clusters d'applications, de vos applications et de vos licences. Assurez-vous que votre environnement répond à ces exigences pour déployer et utiliser Trident Protect.

Compatibilité avec les clusters Kubernetes de Trident Protect

Trident Protect est compatible avec une large gamme 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 Harvester 1.7.0 (ONTAP iSCSI)
- SUSE Rancher
- Gamme VMware Tanzu
- Kubernetes en amont

REMARQUE

- Les sauvegardes Trident Protect sont prises en charge uniquement sur les nœuds de calcul Linux. Les nœuds de calcul Windows ne sont pas pris en charge pour les opérations de sauvegarde.
- Assurez-vous que le cluster sur lequel vous installez Trident Protect est configuré avec un contrôleur de snapshots en cours d'exécution et les CRD associés. Pour installer un contrôleur de snapshots, reportez-vous à la documentation. "ces instructions" .
- Assurez-vous qu'au moins un VolumeSnapshotClass existe. Pour plus d'informations, consultez "VolumeSnapshotClass".
- Helm 4.x ou une version ultérieure est nécessaire pour installer Trident Protect.

Compatibilité du système de stockage Trident Protect

Trident Protect prend en charge les systèmes de stockage 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 Trident 24.02 ou une version plus récente (Trident 24.10 est recommandé).
- 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 les espaces de noms d'application que vous prévoyez d'utiliser pour les applications ou les opérations de gestion des données d'application. 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 échouera.

Conditions requises pour les volumes d'économie nas

Trident Protect prend en charge les opérations de sauvegarde et de restauration sur les volumes NAS économiques. Les snapshots, les clones et la réplication SnapMirror vers des volumes nas-economy ne sont actuellement pas pris en charge. Vous devez activer un répertoire de snapshots pour chaque volume nas-economy que vous prévoyez d'utiliser avec Trident Protect.

REMARQUE

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
```

REMARQUE	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.

Protéger les données avec les machines virtuelles KubeVirt

Trident Protect offre des fonctionnalités de gel et de dégel du système de fichiers pour les machines virtuelles KubeVirt lors des opérations de protection des données afin de garantir la cohérence des données. La méthode de configuration et le comportement par défaut des opérations de gel de VM varient selon les versions de Trident Protect, les versions plus récentes offrant une configuration simplifiée via les paramètres du graphique Helm.

REMARQUE	Pendant les opérations de restauration, tout `VirtualMachineSnapshots` créés pour une machine virtuelle (VM) ne sont pas restaurés.

Trident Protect 25.10 et versions ultérieures

Trident Protect gèle et débloque automatiquement les systèmes de fichiers KubeVirt pendant les opérations de protection des données afin de garantir la cohérence. À partir de Trident Protect 25.10, vous pouvez désactiver ce comportement à l'aide de `vm.freeze` paramètre lors de l'installation du graphique Helm. Ce paramètre est activé par défaut.

```
helm install ... --set vm.freeze=false ...
```

Trident Protect 24.10.1 à 25.06

À partir de Trident Protect 24.10.1, Trident Protect gèle et débloque automatiquement les systèmes de fichiers KubeVirt lors des opérations de protection des données. Vous pouvez désactiver ce comportement automatique à l'aide de la commande suivante :

```
kubectl set env deployment/trident-protect-controller-manager
NEPTUNE_VM_FREEZE=false -n trident-protect
```

Trident Protect 24.10

Trident Protect 24.10 ne garantit pas automatiquement un état cohérent pour les systèmes de fichiers de machines virtuelles KubeVirt lors des opérations de protection des données. Si vous souhaitez protéger les données de votre machine virtuelle KubeVirt à l'aide de Trident Protect 24.10, vous devez activer manuellement la fonctionnalité de gel/dégel des systèmes de fichiers avant l'opération de protection des données. Cela garantit que les systèmes de fichiers sont dans un état cohérent.

Vous pouvez configurer Trident Protect 24.10 pour gérer le gel et le dégel du système de fichiers de la machine virtuelle lors des opérations de protection des données."configuration de la virtualisation" puis en utilisant la commande suivante :

```
kubectl set env deployment/trident-protect-controller-manager
NEPTUNE_VM_FREEZE=true -n trident-protect
```

Pour savoir comment définir des applications et restaurer des machines virtuelles avec Trident Protect, consultez "Protégez les machines virtuelles KubeVirt avec Trident Protect".

Conditions requises pour la réplication SnapMirror

La réplication NetApp SnapMirror est disponible pour une utilisation avec Trident Protect pour les solutions ONTAP suivantes :

- Systèmes NetApp FAS, AFF et ASA sur site. La réplication SnapMirror avec Trident protect n'est actuellement pas prise en charge pour les systèmes ASA r2.
- 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 :

- * NetApp Trident* : NetApp Trident doit exister à la fois sur les clusters Kubernetes source et de destination qui utilisent ONTAP comme backend. Trident Protect prend en charge la réplication avec la technologie NetApp SnapMirror utilisant des classes de stockage reposant sur les pilotes suivants :
  - `ontap-nas` : NFS
  - `ontap-san` : iSCSI
  - `ontap-san` : FC
  - `ontap-san` : NVMe/TCP (nécessite au minimum la version ONTAP 9.15.1)
- **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" .

  À partir de ONTAP 9.10.1, toutes les licences sont livrées sous forme de fichier de licence NetApp (NLF), qui est un fichier unique qui active plusieurs fonctionnalités. Pour plus d'informations, reportez-vous à la

section "Licences incluses avec ONTAP One" .

REMARQUE	Seule la protection asynchrone SnapMirror est prise en charge.

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" .

REMARQUE	S'assurer que les noms de SVM utilisés dans la relation de réplication entre deux clusters ONTAP sont uniques.

- **NetApp Trident et SVM** : les SVM distantes appairées doivent être disponibles pour NetApp Trident sur le cluster de destination.
- **Systèmes de stockage backend gérés** : Vous devez ajouter et gérer des systèmes de stockage backend ONTAP dans Trident Protect pour créer une relation de réplication.

Configuration Trident/ONTAP pour la réplication SnapMirror

Trident Protect exige que vous configuriez au moins un système de stockage dorsal prenant en charge la réplication pour les clusters source et de destination. Si les clusters source et de destination sont identiques, l'application de destination doit utiliser un système de stockage différent de celui de l'application source pour une résilience optimale.

Exigences du cluster Kubernetes pour la réplication SnapMirror

Assurez-vous que vos clusters Kubernetes répondent aux exigences suivantes :

- **Accessibilité AppVault** : les clusters source et de destination doivent disposer d'un accès réseau pour lire et écrire dans AppVault pour la réplication des objets d'application.
- **Connectivité réseau** : configurez les règles de pare-feu, les autorisations de compartiment et les listes d'adresses IP autorisées pour permettre la communication entre les deux clusters et AppVault sur les réseaux WAN.

REMARQUE	De nombreux environnements d'entreprise mettent en œuvre des politiques de pare-feu strictes sur les connexions WAN. Vérifiez ces exigences réseau avec votre équipe d'infrastructure avant de configurer la réplication.

Installez et configurez Trident Protect.

Si votre environnement répond aux exigences de Trident Protect, vous pouvez suivre ces étapes pour installer Trident Protect sur votre cluster. Vous pouvez obtenir Trident Protect auprès 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.

Installer Trident Protect

Installez Trident Protect de NetApp

Étapes

1. Ajout du référentiel Trident Helm :

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

2. Utilisez Helm pour installer Trident Protect. Remplacer `<name-of-cluster>` avec un nom de cluster, qui sera attribué au cluster et utilisé pour identifier les sauvegardes et les instantanés du cluster :

```
helm install trident-protect netapp-trident-protect/trident-protect
--set clusterName=<name-of-cluster> --version 100.2606.0 --create
-namespace --namespace trident-protect
```

3. Pour activer la journalisation de débogage (recommandée pour le dépannage), vous pouvez utiliser la commande suivante (facultative) :

```
helm install trident-protect netapp-trident-protect/trident-protect
--set clusterName=<name-of-cluster> --set logLevel=debug --version
100.2606.0 --create-namespace --namespace trident-protect
```

La journalisation de débogage aide le support NetApp à résoudre les problèmes sans nécessiter de modifications du niveau de journalisation ni de reproduction du problème.

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 crochets par les informations provenant de votre environnement :

Étapes

1. Extrayez les images suivantes sur votre ordinateur local, mettez à jour les balises, puis envoyez-les vers votre registre privé :

```
docker.io/netapp/controller:26.06.0
docker.io/netapp/restic:26.06.0
docker.io/netapp/kopia:26.06.0
docker.io/netapp/kopiablockrestore:26.06.0
docker.io/netapp/trident-autosupport:26.06.0
docker.io/netapp/exechook:26.06.0
docker.io/netapp/resourcebackup:26.06.0
docker.io/netapp/resourcerestore:26.06.0
docker.io/netapp/resourcedelete:26.06.0
docker.io/netapp/trident-protect-utils:v1.0.0
```

Par exemple :

```
docker pull docker.io/netapp/controller:26.06.0
```

```
docker tag docker.io/netapp/controller:26.06.0 <private-registry-url>/controller:26.06.0
```

```
docker push <private-registry-url>/controller:26.06.0
```

REMARQUE Pour obtenir le graphique Helm, téléchargez d'abord le graphique Helm sur une machine ayant accès à Internet à l'aide de `helm pull trident-protect --version 100.2606.0 --repo https://netapp.github.io/trident-protect-helm-chart`, puis copiez le fichier `trident-protect-100.2606.0.tgz` résultant dans votre environnement hors ligne et installez-le en utilisant `helm install trident-protect ./trident-protect-100.2606.0.tgz` au lieu de la référence du dépôt à l'étape finale.

2. Créez l'espace de noms 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` . Assurez-vous qu'il contienne les paramètres Trident Protect suivants :

```
---
imageRegistry: <private-registry-url>
imagePullSecrets:
  - name: regcred
```

REMARQUE

Le `imageRegistry` et `imagePullSecrets` Les valeurs s'appliquent à toutes les images des composants, y compris `resourcebackup` et `resourcerestore` . Si vous envoyez des images vers un chemin de dépôt spécifique au sein de votre registre (par exemple, `example.com:443/my-repo`), veuillez inclure le chemin complet dans le champ d'enregistrement. Cela permettra de garantir que toutes les images sont extraites de `<private-registry-url>/<image-name>:<tag>`.

7. Utilisez Helm pour installer Trident Protect. Remplacer `<name_of_cluster>` avec un nom de cluster, qui sera attribué au cluster et utilisé pour identifier les sauvegardes et les instantanés du cluster :

```
helm install trident-protect netapp-trident-protect/trident-protect
--set clusterName=<name_of_cluster> --version 100.2606.0 --create
-namespace --namespace trident-protect -f protectValues.yaml
```

8. Pour activer la journalisation de débogage (recommandée pour le dépannage), vous pouvez utiliser la commande suivante (facultative) :

```
helm install trident-protect netapp-trident-protect/trident-protect
--set clusterName=<name-of-cluster> --set logLevel=debug --version
100.2606.0 --create-namespace --namespace trident-protect -f
protectValues.yaml
```

La journalisation de débogage aide le support NetApp à résoudre les problèmes sans nécessiter de modifications du niveau de journalisation ni de reproduction du problème.

REMARQUE	Pour plus d'options de configuration des graphiques Helm, notamment les paramètres AutoSupport et le filtrage des espaces de noms, veuillez consulter la documentation. "Personnaliser l'installation de Trident Protect".

Installez le plugin CLI Trident Protect

Vous pouvez utiliser le plugin en ligne de commande Trident Protect, qui est une extension de Trident. `tridentctl` utilitaire, pour créer et interagir avec les ressources personnalisées (CR) de Trident Protect.

Installez le plugin CLI Trident Protect

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

Étapes

1. Téléchargez le plugin CLI Trident Protect :

```
curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-protect/releases/download/26.06.0/tridentctl-protect-linux-amd64
```

Télécharger le plug-in pour les processeurs Linux ARM64

Étapes

1. Téléchargez le plugin CLI Trident Protect :

```
curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-protect/releases/download/26.06.0/tridentctl-protect-linux-arm64
```

Télécharger le plug-in pour les processeurs Mac AMD64

Étapes

1. Téléchargez le plugin CLI Trident Protect :

```
curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-protect/releases/download/26.06.0/tridentctl-protect-macos-amd64
```

Télécharger le plug-in pour les processeurs Mac ARM64

Étapes

1. Téléchargez le plugin CLI Trident Protect :

```
curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-protect/releases/download/26.06.0/tridentctl-protect-macos-arm64
```

1. Activer les autorisations d'exécution pour le binaire du plug-in :

```
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 du plug-in vers un emplacement de votre répertoire personnel. Dans ce cas, il est recommandé de s'assurer que l'emplacement fait partie de votre variable PATH :

```
cp ./tridentctl-protect ~/bin/
```

REMARQUE La copie du plug-in vers un emplacement de la variable PATH vous permet d'utiliser le plug-in en tapant `tridentctl-protect` ou `tridentctl protect` à partir de n'importe quel emplacement.

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 le plugin CLI Trident Protect installé, vous pouvez activer la saisie semi-automatique pour certaines commandes.

Activer la saisie semi-automatique pour le shell Bash

Étapes

1. Créez le script de complétion :

```
tridentctl-protect completion bash > 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.

Étapes

1. Créez le script de complétion :

```
tridentctl-protect completion zsh > 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.

Personnaliser l'installation de Trident Protect

Vous pouvez personnaliser la configuration par défaut de Trident Protect pour répondre aux exigences spécifiques de votre environnement.

Spécifiez les limites de ressources du conteneur Trident Protect

Vous pouvez utiliser un fichier de configuration pour spécifier les limites de ressources des conteneurs Trident Protect après l'installation de Trident Protect. La définition de limites de ressources vous permet de contrôler la quantité de ressources du cluster consommées par les opérations de Trident Protect.

Étapes

1. Créez un fichier nommé `resourceLimits.yaml`.
2. Renseignez le fichier avec les options de limite de ressources pour les conteneurs Trident Protect en fonction des besoins de votre environnement.

   L'exemple de fichier de configuration suivant montre les paramètres disponibles et contient les valeurs par défaut pour chaque limite de ressource :

```
---
jobResources:
  defaults:
    limits:
      cpu: 8000m
      memory: 10000Mi
      ephemeralStorage: ""
    requests:
      cpu: 100m
      memory: 100Mi
      ephemeralStorage: ""
  resticVolumeBackup:
    limits:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
    requests:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
  resticVolumeRestore:
    limits:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
```

```
    requests:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
  kopiaVolumeBackup:
    limits:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
    requests:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
  kopiaVolumeRestore:
    limits:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
    requests:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
```

3. Appliquer les valeurs du `resourceLimits.yaml` fichier :

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect -f resourceLimits.yaml --reuse-values
```

Personnaliser les contraintes de contexte de sécurité

Vous pouvez utiliser un fichier de configuration pour modifier les contraintes de contexte de sécurité OpenShift (SCC) pour les conteneurs Trident Protect après avoir installé Trident Protect. Ces contraintes définissent les restrictions de sécurité des pods dans un cluster Red Hat OpenShift.

Étapes

1. Créez un fichier nommé `sccconfig.yaml`.
2. Ajoutez l'option SCC au fichier et modifiez les paramètres en fonction des besoins de votre environnement.

   L'exemple suivant montre les valeurs par défaut des paramètres de l'option SCC :

```
scc:
  create: true
  name: trident-protect-job
  priority: 1
```

Ce tableau décrit les paramètres de l'option SCC :

Paramètre	Description	Valeur par défaut
création	Détermine si une ressource SCC peut être créée. Une ressource SCC ne sera créée que si est défini sur `true` et que `scc.create` le processus d'installation de Helm identifie un environnement OpenShift. Si ne fonctionne pas sur OpenShift ou si `scc.create` est défini sur `false`, aucune ressource SCC ne sera créée.	vrai
nom	Spécifie le nom du SCC.	travail-protection-Trident
priorité	Définit la priorité du SCC. Les SCC ayant des valeurs de priorité plus élevées sont évalués avant ceux ayant des valeurs plus faibles.	1

3. Appliquer les valeurs du `sccconfig.yaml` fichier :

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect -f sccconfig.yaml --reuse-values
```

Les valeurs par défaut seront remplacées par celles spécifiées dans le `sccconfig.yaml` fichier.

Configurer les paramètres supplémentaires du graphique de barre de Trident Protect

Vous pouvez personnaliser les paramètres AutoSupport, le filtrage des espaces de noms et la gestion des certificats webhook OpenShift pour répondre à vos besoins spécifiques. Le tableau suivant décrit les paramètres de configuration disponibles :

Paramètre	Type	Description
autoSupport.proxy	chaîne	Configure une URL proxy pour les connexions NetApp AutoSupport . Utilisez ceci pour acheminer les téléchargements de bundles de support via un serveur proxy. Exemple: `http://my.proxy.url` .

Paramètre	Type	Description
autoSupport.insecure	booléen	Ignore la vérification TLS pour les connexions proxy AutoSupport lorsqu'elle est définie sur `true` . À utiliser uniquement pour les connexions proxy non sécurisées. (défaut: `false`)
autoSupport.activé	booléen	Active ou désactive les téléchargements quotidiens des modules Trident Protect AutoSupport . Lorsqu'il est réglé sur `false` Les téléchargements quotidiens programmés sont désactivés, mais vous pouvez toujours générer manuellement des ensembles de support. (défaut: `true`)
restaurerSkipNamespaceAnnotations	chaîne	Liste séparée par des virgules d'annotations d'espace de noms à exclure des opérations de sauvegarde et de restauration. Vous permet de filtrer les espaces de noms en fonction des annotations.
restaurerSkipNamespaceLabels	chaîne	Liste séparée par des virgules d'étiquettes d'espace de noms à exclure des opérations de sauvegarde et de restauration. Vous permet de filtrer les espaces de noms en fonction des étiquettes.
useOpenshiftServiceCA	booléen	Permet l'intégration avec l'opérateur d'autorité de certification de service natif de OpenShift pour la création et l'injection automatiques de certificats TLS pour le webhook d'admission Trident Protect. Lorsqu'il est défini sur `true`, l'opérateur d'autorité de certification de service gère automatiquement les certificats TLS du webhook. À utiliser uniquement sur les clusters OpenShift. (par défaut : `false`)

Vous pouvez configurer ces options à l'aide d'un fichier de configuration YAML ou d'indicateurs de ligne de commande :

Utiliser le fichier YAML

Étapes

1. Créez un fichier de configuration et nommez-le `values.yaml` .
2. Dans le fichier que vous avez créé, ajoutez les options de configuration que vous souhaitez personnaliser.

```
autoSupport:
  enabled: false
  proxy: http://my.proxy.url
  insecure: true
restoreSkipNamespaceAnnotations: "annotation1,annotation2"
restoreSkipNamespaceLabels: "label1,label2"
useOpenshiftServiceCA: true
```

3. Après avoir rempli le `values.yaml` fichier avec les valeurs correctes, appliquez le fichier de configuration :

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect -f values.yaml --reuse-values
```

Utiliser l'indicateur CLI

Étapes

1. Utilisez la commande suivante avec le `--set` indicateur pour spécifier des paramètres individuels :

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect \
  --set autoSupport.enabled=false \
  --set autoSupport.proxy=http://my.proxy.url \
  --set-string
restoreSkipNamespaceAnnotations="{annotation1,annotation2}" \
  --set-string restoreSkipNamespaceLabels="{label1,label2}" \
  --set useOpenshiftServiceCA=true \
  --reuse-values
```

Limiter les pods Trident Protect à des nœuds spécifiques

Vous pouvez utiliser la contrainte de sélection de nœuds nodeSelector de Kubernetes pour contrôler quels nœuds sont éligibles pour exécuter des pods Trident Protect, en fonction des étiquettes de nœud. Par défaut, Trident Protect est limité aux nœuds exécutant Linux. Vous pouvez personnaliser davantage ces contraintes en fonction de vos besoins.

Étapes

1. Créez un fichier nommé `nodeSelectorConfig.yaml`.
2. Ajoutez l'option nodeSelector au fichier et modifiez le fichier pour ajouter ou modifier des libellés de nœud afin de les restreindre en fonction des besoins de votre environnement. Par exemple, le fichier suivant contient la restriction par défaut du système d'exploitation, mais cible également une région et un nom d'application spécifiques :

```
nodeSelector:
  kubernetes.io/os: linux
  region: us-west
  app.kubernetes.io/name: mysql
```

3. Appliquer les valeurs du `nodeSelectorConfig.yaml` fichier :

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect -f nodeSelectorConfig.yaml --reuse-values
```

Ceci remplace les restrictions par défaut par celles que vous avez spécifiées dans le `nodeSelectorConfig.yaml` fichier.

Gérer Trident Protect

Gérer l'autorisation et le contrôle d'accès de Trident Protect

Trident Protect utilise le modèle Kubernetes de contrôle d'accès basé sur les rôles (RBAC). Par défaut, Trident Protect fournit un seul espace de noms système et son compte de service par défaut associé. Si votre organisation compte de nombreux utilisateurs ou des besoins de sécurité spécifiques, vous pouvez utiliser les fonctionnalités RBAC de Trident Protect pour obtenir un contrôle plus précis 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).

REMARQUE	Seuls les administrateurs devraient avoir accès aux objets de ressources personnalisés privilégiés de Trident Protect, notamment : • **AppVault** : nécessite les données d'informations d'identification du compartiment • **AutoSupportBundle** : Collecte les métriques, les journaux et 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 lors de l'installation de 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.

Surveiller les ressources de Trident Protect

Vous pouvez utiliser les outils open source kube-state-metrics, Prometheus et Alertmanager pour surveiller l'état des ressources protégées par Trident Protect.

Le service kube-state-metrics génère des métriques à partir des communications de l'API Kubernetes. Son utilisation conjointe avec Trident Protect permet d'obtenir des informations utiles sur l'état des ressources de votre environnement.

Prometheus est une boîte à outils capable d'ingérer les données générées par kube-state-metrics et de les présenter sous forme d'informations facilement lisibles sur ces objets. Ensemble, kube-state-metrics et Prometheus vous permettent de surveiller l'état et la santé des ressources que vous gérez avec Trident Protect.

AlertManager est un service qui ingère les alertes envoyées par des outils tels que Prometheus et les redirige vers les destinations que vous configurez.

REMARQUE	Les configurations et les conseils inclus dans ces étapes ne sont que des exemples ; vous devez les personnaliser en fonction de votre environnement. Reportez-vous à la documentation officielle suivante pour obtenir des instructions et une assistance spécifiques : • "documentation sur les indicateurs d'état kube" • "Documentation Prometheus" • "Documentation d'AlertManager"

Étape 1 : installez les outils de surveillance

Pour activer la surveillance des ressources dans Trident Protect, vous devez installer et configurer kube-state-metrics, Promethus et Alertmanager.

Installez les indicateurs kube-state

Vous pouvez installer kube-state-metrics à l'aide de Helm.

Étapes

1. Ajoutez le graphique Helm kube-state-metrics. Par exemple :

```
helm repo add prometheus-community https://prometheus-
community.github.io/helm-charts
helm repo update
```

2. Appliquez le CRD Prometheus ServiceMonitor au cluster :

```
kubectl apply -f https://raw.githubusercontent.com/prometheus-
operator/prometheus-operator/main/example/prometheus-operator-
crd/monitoring.coreos.com_servicemonitors.yaml
```

3. Créez un fichier de configuration pour le graphique Helm (par exemple, `metrics-config.yaml`). Vous pouvez personnaliser l'exemple de configuration suivant en fonction de votre environnement :

Metrics-config.yaml : configuration du graphique Helm kube-state-metrics

```
---
extraArgs:
  # Collect only custom metrics
  - --custom-resource-state-only=true

customResourceState:
  enabled: true
  config:
    kind: CustomResourceStateMetrics
    spec:
      resources:
      - groupVersionKind:
          group: protect.trident.netapp.io
          kind: "Backup"
          version: "v1"
        labelsFromPath:
          backup_uid: [metadata, uid]
          backup_name: [metadata, name]
          creation_time: [metadata, creationTimestamp]
        metrics:
        - name: backup_info
          help: "Exposes details about the Backup state"
          each:
            type: Info
            info:
              labelsFromPath:
                appVaultReference: ["spec", "appVaultRef"]
                appReference: ["spec", "applicationRef"]
rbac:
  extraRules:
  - apiGroups: ["protect.trident.netapp.io"]
    resources: ["backups"]
    verbs: ["list", "watch"]

# Collect metrics from all namespaces
namespaces: ""

# Ensure that the metrics are collected by Prometheus
prometheus:
  monitor:
    enabled: true
```

4. Installez les indicateurs d'état kube en déployant le graphique Helm. Par exemple :

```
helm install custom-resource -f metrics-config.yaml prometheus-
community/kube-state-metrics --version 5.21.0
```

5. Configurez kube-state-metrics pour générer des métriques pour les ressources personnalisées utilisées par Trident Protect en suivant les instructions du guide. "documentation sur les ressources personnalisées kube-state-metrics" .

Installez Prometheus

Vous pouvez installer Prometheus en suivant les instructions de la "Documentation Prometheus".

Installez AlertManager

Vous pouvez installer AlertManager en suivant les instructions de la "Documentation d'AlertManager".

Étape 2 : configurer les outils de surveillance pour qu'ils fonctionnent ensemble

Après avoir installé les outils de surveillance, vous devez les configurer pour qu'ils fonctionnent ensemble.

Étapes

1. Intégrez des metrics kube-state avec Prometheus. Modifiez le fichier de configuration Prometheus (`prometheus.yaml`) et ajoutez les informations du service kube-state-metrics. Par exemple :

prometheus.yaml : intégration du service kube-state-metrics avec Prometheus

```
---
apiVersion: v1
kind: ConfigMap
metadata:
  name: prometheus-config
  namespace: trident-protect
data:
  prometheus.yaml: |
    global:
      scrape_interval: 15s
    scrape_configs:
      - job_name: 'kube-state-metrics'
        static_configs:
          - targets: ['kube-state-metrics.trident-protect.svc:8080']
```

2. Configurez Prometheus pour acheminer les alertes vers AlertManager. Modifiez le fichier de configuration Prometheus (`prometheus.yaml`) et ajoutez la section suivante :

prometheus.yaml : envoyer des alertes à Alertmanager

```
alerting:
  alertmanagers:
    - static_configs:
        - targets:
            - alertmanager.trident-protect.svc:9093
```

Résultat

Prometheus peut désormais collecter des metrics à partir de metrics kube-state et envoyer des alertes à AlertManager. Vous êtes maintenant prêt à configurer les conditions qui déclenchent une alerte et l'emplacement où les alertes doivent être envoyées.

Étape 3 : configuration des alertes et des destinations d'alertes

Une fois que vous avez configuré les outils pour qu'ils fonctionnent ensemble, vous devez configurer le type d'informations qui déclenche des alertes et l'emplacement où elles doivent être envoyées.

Exemple d'alerte : échec de la sauvegarde

L'exemple suivant définit une alerte critique qui est déclenchée lorsque l'état de la ressource personnalisée de sauvegarde est défini sur `Error` pendant 5 secondes ou plus. Vous pouvez personnaliser cet exemple pour l'adapter à votre environnement et inclure cet extrait YAML dans votre `prometheus.yaml` fichier de configuration :

rules.yaml : définir une alerte Prometheus pour les sauvegardes ayant échoué

```
rules.yaml: |
  groups:
    - name: fail-backup
        rules:
          - alert: BackupFailed
            expr: kube_customresource_backup_info{status="Error"}
            for: 5s
            labels:
              severity: critical
            annotations:
              summary: "Backup failed"
              description: "A backup has failed."
```

Configurez AlertManager pour envoyer des alertes à d'autres canaux

Vous pouvez configurer AlertManager pour envoyer des notifications à d'autres canaux, tels que les e-mails, PagerDuty, Microsoft Teams ou d'autres services de notification en spécifiant la configuration respective dans le `alertmanager.yaml` fichier.

L'exemple suivant configure AlertManager pour envoyer des notifications à un canal Slack. Pour personnaliser cet exemple en fonction de votre environnement, remplacez la valeur de la `api_url` clé par l'URL Slack webhook utilisée dans votre environnement :

alertmanager.yaml : envoyer des alertes à un canal Slack

```
data:
  alertmanager.yaml: |
    global:
      resolve_timeout: 5m
    route:
      receiver: 'slack-notifications'
    receivers:
      - name: 'slack-notifications'
        slack_configs:
          - api_url: '<your-slack-webhook-url>'
            channel: '#failed-backups-channel'
            send_resolved: false
```

Générer un ensemble de support Trident Protect

Trident Protect permet aux administrateurs de générer des ensembles contenant des informations utiles au support NetApp , notamment des journaux, des métriques et des informations de topologie sur les clusters et les applications gérés. Si vous êtes connecté à Internet, vous pouvez télécharger des bundles de support sur le site de support NetApp (NSS) à l'aide d'un fichier de ressources personnalisé (CR).

Créez un bundle de support à l'aide d'une CR

Étapes

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. Après avoir rempli le `trident-protect-support-bundle.yaml` fichier avec les valeurs correctes, appliquez le CR :

```
kubectl apply -f trident-protect-support-bundle.yaml -n trident-protect
```

Créez un bundle de support à l'aide de l'interface de ligne de commande

Étapes

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> -n trident-protect
```

Surveiller et récupérer le pack de support

Après avoir créé un bundle de support à l'aide de l'une ou l'autre méthode, vous pouvez surveiller sa progression de génération et le récupérer sur votre système local.

Étapes

1. Attendez le `status.generationState` atteindre `Completed` État. Vous pouvez surveiller la progression de la génération avec la commande suivante :

   ```
   kubectl get autosupportbundle trident-protect-support-bundle -n trident-protect
   ```

2. Récupérez le pack de support sur votre système local. Obtenez la commande de copie à partir du bundle AutoSupport terminé :

   ```
   kubectl describe autosupportbundle trident-protect-support-bundle -n trident-protect
   ```

   Trouver le `kubectl cp` commande à partir de la sortie et exécutez-la, en remplaçant l'argument de destination par votre répertoire local préféré.

Amélioration de Trident Protect

Vous pouvez mettre à jour Trident Protect vers la dernière version pour bénéficier des nouvelles fonctionnalités ou des correctifs de bugs.

REMARQUE

- Lors d'une mise à niveau depuis la version 24.10, les snapshots exécutés pendant la mise à niveau peuvent échouer. Cet échec n'empêche pas la création de futurs snapshots, qu'ils soient manuels ou planifiés. Si un snapshot échoue pendant la mise à niveau, vous pouvez en créer un manuellement pour garantir la protection de votre application.

  Pour éviter d'éventuels échecs, vous pouvez désactiver toutes les planifications d'instantanés avant la mise à niveau et les réactiver ensuite. Cependant, cela entraînera l'absence d'instantanés planifiés pendant la période de mise à niveau.

- Pour les installations de registre privé, assurez-vous que le graphique Helm et les images requis pour la version cible sont disponibles dans votre registre privé et vérifiez que vos valeurs Helm personnalisées sont compatibles avec la nouvelle version du graphique. Pour plus d'informations, veuillez consulter"Installez Trident Protect à partir d'un registre privé" .

Étape 1 : sélectionnez une version

Les versions de Trident Protect suivent une convention de nommage basée sur la date `YY.MM`, où « YY » est les deux derniers chiffres de l'année et « MM » est le mois. Les versions dot suivent une `YY.MM.X` convention, où « X » est le niveau de correctif. Vous sélectionnerez la version vers laquelle effectuer la mise 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 un intervalle de quatre versions maximum par rapport à votre version installée. Par exemple, vous pouvez effectuer une mise à niveau directe de 24.10 (ou de toute version dérivée de 24.10) vers 25.10.
- Si vous effectuez une mise à niveau depuis une version en dehors de la fenêtre de quatre versions, effectuez une mise à niveau en plusieurs étapes. Utilisez les instructions de mise à niveau pour la "version antérieure" version à partir de laquelle vous effectuez la mise à niveau afin de passer à la version la plus récente qui correspond à la fenêtre de quatre versions. Par exemple, si vous utilisez la version 24.10 et souhaitez passer à la version 26.02 :
  a. Première mise à jour de 24.10 à 25.02.
  b. Ensuite, passez de 25.02 à 26.02.

Étape 2 : Mise à niveau de Trident Protect

Pour mettre à niveau Trident Protect, procédez comme suit.

Étapes

1. Mettez à jour le référentiel Trident Helm :

```
helm repo update
```

2. Mettez à niveau les CRD Trident Protect :

REMARQUE

Cette étape est nécessaire si vous effectuez une mise à niveau depuis une version antérieure à la 25.06, car les CRD sont désormais inclus dans le tableau Trident Protect Helm.

   a. Exécutez cette commande pour déplacer la gestion des CRD de `trident-protect-crds` à

`trident-protect` :

```
kubectl get crd | grep protect.trident.netapp.io | awk '{print $1}' |
xargs -I {} kubectl patch crd {} --type merge -p '{"metadata":
{"annotations":{"meta.helm.sh/release-name": "trident-protect"}}}'
```

b. Exécutez cette commande pour supprimer le secret Helm pour le `trident-protect-crds` graphique:

REMARQUE Ne désinstallez pas le `trident-protect-crds` graphique à l'aide de Helm, car cela pourrait supprimer vos CRD et toutes les données associées.

```
kubectl delete secret -n trident-protect -l name=trident-protect-
crds,owner=helm
```

3. Mise à niveau de Trident Protect :

```
helm upgrade trident-protect netapp-trident-protect/trident-protect
--version 100.2606.0 --namespace trident-protect
```

REMARQUE Vous pouvez configurer le niveau de journalisation lors de la mise à niveau en ajoutant `--set logLevel=debug` à la commande de mise à niveau. Le niveau de journalisation par défaut est `warn` . L'activation de la journalisation de débogage est recommandée pour le dépannage, car elle aide le support NetApp à diagnostiquer les problèmes sans nécessiter de modifications du niveau de journalisation ni de reproduction du problème.

Gérez et protégez les applications

Utilisez les objets Trident Protect AppVault pour gérer les compartiments.

La ressource personnalisée (CR) du compartiment pour Trident Protect est connue sous le nom d'AppVault. Les objets AppVault sont la représentation déclarative du flux de travail Kubernetes d'un bucket de stockage. Un AppVault CR contient les configurations nécessaires pour qu'un bucket soit utilisé 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 AppVaults.

Vous devez créer une CR AppVault manuellement ou depuis la ligne de commande lorsque vous effectuez des opérations de protection des données sur une application. La CR AppVault est spécifique à votre environnement et vous pouvez vous inspirer des exemples de cette page pour créer des CR AppVault.

REMARQUE	Assurez-vous que le fichier CR d'AppVault se trouve sur le cluster où Trident Protect est installé. Si le CR AppVault n'existe pas ou si vous ne pouvez pas y accéder, la ligne de commande affiche une erreur.

Configurer l'authentification et les mots de passe AppVault

Avant de créer un AppVault CR, assurez-vous que l'AppVault et le dispositif de transfert de données que vous choisissez peuvent s'authentifier auprès du fournisseur et de toutes les ressources associées.

Mots de passe du référentiel de Data Mover

Lorsque vous créez des objets AppVault à l'aide de CR ou du plugin CLI Trident Protect, vous pouvez spécifier un secret Kubernetes avec des mots de passe personnalisés pour le chiffrement Restic et Kopia. Si vous ne spécifiez pas de mot de passe secret, Trident Protect utilise un mot de passe par défaut.

- Lors de la création manuelle de CR AppVault, utilisez le champ **spec.dataMoverPasswordSecretRef** pour spécifier le secret.
- Lors de la création d'objets AppVault à l'aide de l'interface de ligne de commande Trident Protect, utilisez le `--data-mover-password-secret-ref` argument permettant de préciser le secret.

Créez un mot de passe secret pour le référentiel du Data Mover

Utilisez les exemples suivants pour créer le mot de passe secret. Lorsque vous créez des objets AppVault, vous pouvez indiquer à Trident Protect d'utiliser ce secret pour s'authentifier auprès du référentiel de transfert de données.

REMARQUE	• Selon le mécanisme de déplacement des données que vous utilisez, il vous suffit d'inclure le mot de passe correspondant à ce mécanisme de transfert de données. Par exemple, si vous utilisez Restic et que vous ne prévoyez pas d'utiliser Kopia à l'avenir, vous ne pouvez inclure que le mot de passe Restic lorsque vous créez le secret. • Conservez le mot de passe en lieu sûr. Vous en aurez besoin pour restaurer les données sur le même cluster ou sur un autre. Si le cluster ou le `trident-protect` l'espace de noms est supprimé, vous ne pourrez pas restaurer vos sauvegardes ou instantanés sans le mot de passe.

Utiliser une CR

```
---
apiVersion: v1
data:
  KOPIA_PASSWORD: <base64-encoded-password>
  RESTIC_PASSWORD: <base64-encoded-password>
kind: Secret
metadata:
  name: my-optional-data-mover-secret
  namespace: trident-protect
type: Opaque
```

Utiliser l'interface de ligne de commande

```
kubectl create secret generic my-optional-data-mover-secret \
--from-literal=KOPIA_PASSWORD=<plain-text-password> \
--from-literal=RESTIC_PASSWORD=<plain-text-password> \
-n trident-protect
```

Autorisations IAM de stockage compatibles S3

Lorsque vous accédez à un stockage compatible S3 tel qu'Amazon S3, Generic S3, "StorageGRID S3" , ou "ONTAP S3" Lors de l'utilisation de Trident Protect, vous devez vous assurer que les informations d'identification de l'utilisateur que vous fournissez disposent des autorisations nécessaires pour accéder au compartiment. Voici un exemple de politique accordant les autorisations minimales requises pour l'accès avec Trident Protect. Vous pouvez appliquer cette politique à l'utilisateur qui gère les politiques de compartiment compatibles S3.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:GetObject",
        "s3:ListBucket",
        "s3:DeleteObject"
      ],
      "Resource": "*"
    }
  ]
}
```

Pour plus d'informations sur les politiques Amazon S3, reportez-vous aux exemples dans le "Documentation Amazon S3" .

Identité du pod EKS pour l'authentification Amazon S3 (AWS)

Trident Protect prend en charge EKS Pod Identity pour les opérations de déplacement de données Kopia. Cette fonctionnalité permet un accès sécurisé aux buckets S3 sans stocker les informations d'identification AWS dans les secrets Kubernetes.

Configuration requise pour l'identification du pod EKS avec Trident Protect

Avant d'utiliser EKS Pod Identity avec Trident Protect, assurez-vous des points suivants :

- L'identité de pod est activée sur votre cluster EKS.
- Vous avez créé un rôle IAM avec les autorisations de compartiment S3 nécessaires. Pour en savoir plus, consultez"Autorisations IAM de stockage compatibles S3" .
- Le rôle IAM est associé aux comptes de service Trident Protect suivants :
  - `<trident-protect>-controller-manager`
  - `<trident-protect>-resource-backup`
  - `<trident-protect>-resource-restore`
  - `<trident-protect>-resource-delete`

Pour obtenir des instructions détaillées sur l'activation de l'identité du pod et l'association des rôles IAM aux comptes de service, reportez-vous à la "Documentation sur l'identité des pods AWS EKS" .

Configuration AppVault Lorsque vous utilisez l'identité du pod EKS, configurez votre AppVault CR avec le `useIAM: true` drapeau au lieu d'informations d'identification explicites :

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: eks-protect-vault
  namespace: trident-protect
spec:
  providerType: AWS
  providerConfig:
    s3:
      bucketName: trident-protect-aws
      endpoint: s3.example.com
      useIAM: true
```

Exemples de génération de clés AppVault pour les fournisseurs cloud

Lors de la définition d'un AppVault CR, vous devez inclure des informations d'identification pour accéder aux ressources hébergées par le fournisseur, sauf si vous utilisez l'authentification IAM. La manière dont vous générez les clés pour les informations d'identification varie selon le fournisseur. Voici des exemples de génération de clés en ligne de commande pour plusieurs fournisseurs. Vous pouvez utiliser les exemples suivants pour créer des clés pour les informations d'identification de chaque fournisseur de cloud.

Google Cloud

```
kubectl create secret generic <secret-name> \
--from-file=credentials=<mycreds-file.json> \
-n trident-protect
```

Amazon S3 (AWS)

```
kubectl create secret generic <secret-name> \
--from-literal=accessKeyID=<objectstorage-accesskey> \
--from-literal=secretAccessKey=<amazon-s3-trident-protect-src-bucket
-secret> \
-n trident-protect
```

Microsoft Azure

```
kubectl create secret generic <secret-name> \
--from-literal=accountKey=<secret-name> \
-n trident-protect
```

S3 générique

```
kubectl create secret generic <secret-name> \
--from-literal=accessKeyID=<objectstorage-accesskey> \
--from-literal=secretAccessKey=<generic-s3-trident-protect-src-bucket
-secret> \
-n trident-protect
```

ONTAP S3

```
kubectl create secret generic <secret-name> \
--from-literal=accessKeyID=<objectstorage-accesskey> \
--from-literal=secretAccessKey=<ontap-s3-trident-protect-src-bucket
-secret> \
-n trident-protect
```

StorageGRID S3

```
kubectl create secret generic <secret-name> \
--from-literal=accessKeyID=<objectstorage-accesskey> \
--from-literal=secretAccessKey=<storagegrid-s3-trident-protect-src
-bucket-secret> \
-n trident-protect
```

Exemples de création d'AppVault

Voici des exemples de définitions AppVault pour chaque fournisseur.

Exemples de CR AppVault

Vous pouvez utiliser les exemples CR suivants pour créer des objets AppVault pour chaque fournisseur de cloud.

REMARQUE

- Vous pouvez facultativement spécifier un code secret Kubernetes qui contient des mots de passe personnalisés pour le chiffrement du référentiel Restic et Kopia. Pour plus d'informations, reportez-vous à la section Mots de passe du référentiel de Data Mover .
- Pour les objets AppVault Amazon S3 (AWS), vous pouvez spécifier un jeton de session, utile si vous utilisez l'authentification SSO. Ce jeton est créé lorsque vous générez des clés pour le fournisseur dans Exemples de génération de clés AppVault pour les fournisseurs cloud.
- Pour les objets S3 AppVault, vous pouvez spécifier une URL proxy de sortie pour le trafic S3 sortant à l'aide de la `spec.providerConfig.S3.proxyURL` clé.

Google Cloud

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: gcp-trident-protect-src-bucket
  namespace: trident-protect
spec:
  dataMoverPasswordSecretRef: my-optional-data-mover-secret
  providerType: GCP
  providerConfig:
    gcp:
      bucketName: trident-protect-src-bucket
      projectID: project-id
  providerCredentials:
    credentials:
      valueFromSecret:
        key: credentials
        name: gcp-trident-protect-src-bucket-secret
```

Amazon S3 (AWS)

```yaml
---
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: amazon-s3-trident-protect-src-bucket
  namespace: trident-protect
spec:
  dataMoverPasswordSecretRef: my-optional-data-mover-secret
  providerType: AWS
  providerConfig:
    s3:
      bucketName: trident-protect-src-bucket
      endpoint: s3.example.com
      proxyURL: http://10.1.1.1:3128
  providerCredentials:
    accessKeyID:
      valueFromSecret:
        key: accessKeyID
        name: s3-secret
    secretAccessKey:
      valueFromSecret:
        key: secretAccessKey
        name: s3-secret
    sessionToken:
      valueFromSecret:
        key: sessionToken
        name: s3-secret
```

REMARQUE	Pour les environnements EKS utilisant Pod Identity avec le déplaceur de données Kopia, vous pouvez supprimer le `providerCredentials` section et ajouter `useIAM: true` sous le `s3` configuration à la place.

Microsoft Azure

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: azure-trident-protect-src-bucket
  namespace: trident-protect
spec:
  dataMoverPasswordSecretRef: my-optional-data-mover-secret
  providerType: Azure
  providerConfig:
    azure:
      accountName: account-name
      bucketName: trident-protect-src-bucket
  providerCredentials:
    accountKey:
      valueFromSecret:
        key: accountKey
        name: azure-trident-protect-src-bucket-secret
```

S3 générique

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: generic-s3-trident-protect-src-bucket
  namespace: trident-protect
spec:
  dataMoverPasswordSecretRef: my-optional-data-mover-secret
  providerType: GenericS3
  providerConfig:
    s3:
      bucketName: trident-protect-src-bucket
      endpoint: s3.example.com
      proxyURL: http://10.1.1.1:3128
  providerCredentials:
    accessKeyID:
      valueFromSecret:
        key: accessKeyID
        name: s3-secret
    secretAccessKey:
      valueFromSecret:
        key: secretAccessKey
        name: s3-secret
```

ONTAP S3

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: ontap-s3-trident-protect-src-bucket
  namespace: trident-protect
spec:
  dataMoverPasswordSecretRef: my-optional-data-mover-secret
  providerType: OntapS3
  providerConfig:
    s3:
      bucketName: trident-protect-src-bucket
      endpoint: s3.example.com
      proxyURL: http://10.1.1.1:3128
  providerCredentials:
    accessKeyID:
      valueFromSecret:
        key: accessKeyID
        name: s3-secret
    secretAccessKey:
      valueFromSecret:
        key: secretAccessKey
        name: s3-secret
```

StorageGRID S3

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: storagegrid-s3-trident-protect-src-bucket
  namespace: trident-protect
spec:
  dataMoverPasswordSecretRef: my-optional-data-mover-secret
  providerType: StorageGridS3
  providerConfig:
    s3:
      bucketName: trident-protect-src-bucket
      endpoint: s3.example.com
      proxyURL: http://10.1.1.1:3128
  providerCredentials:
    accessKeyID:
      valueFromSecret:
        key: accessKeyID
        name: s3-secret
    secretAccessKey:
      valueFromSecret:
        key: secretAccessKey
        name: s3-secret
```

Exemples de création d'AppVault à l'aide de l'interface de ligne de commande Trident Protect

Vous pouvez utiliser les exemples de commandes CLI suivants pour créer AppVault CRS pour chaque fournisseur.

REMARQUE

- Vous pouvez facultativement spécifier un code secret Kubernetes qui contient des mots de passe personnalisés pour le chiffrement du référentiel Restic et Kopia. Pour plus d'informations, reportez-vous à la section Mots de passe du référentiel de Data Mover .
- Pour les objets S3 AppVault, vous pouvez spécifier une URL de sortie proxy pour le trafic S3 sortant à l'aide de l' `--proxy-url <ip_address:port>`argument.

Google Cloud

```
tridentctl-protect create vault GCP <vault-name> \
--bucket <mybucket> \
--project <my-gcp-project> \
--secret <secret-name>/credentials \
--data-mover-password-secret-ref <my-optional-data-mover-secret> \
-n trident-protect
```

Amazon S3 (AWS)

```
tridentctl-protect create vault AWS <vault-name> \
--bucket <bucket-name> \
--secret  <secret-name>  \
--endpoint <s3-endpoint> \
--data-mover-password-secret-ref <my-optional-data-mover-secret> \
-n trident-protect
```

Microsoft Azure

```
tridentctl-protect create vault Azure <vault-name> \
--account <account-name> \
--bucket <bucket-name> \
--secret <secret-name> \
--data-mover-password-secret-ref <my-optional-data-mover-secret> \
-n trident-protect
```

S3 générique

```
tridentctl-protect create vault GenericS3 <vault-name> \
--bucket <bucket-name> \
--secret  <secret-name>  \
--endpoint <s3-endpoint> \
--data-mover-password-secret-ref <my-optional-data-mover-secret> \
-n trident-protect
```

ONTAP S3

```
tridentctl-protect create vault OntapS3 <vault-name> \
--bucket <bucket-name> \
--secret  <secret-name>  \
--endpoint <s3-endpoint> \
--data-mover-password-secret-ref <my-optional-data-mover-secret> \
-n trident-protect
```

StorageGRID S3

```
tridentctl-protect create vault StorageGridS3 <vault-name> \
--bucket <bucket-name> \
--secret  <secret-name>  \
--endpoint <s3-endpoint> \
--data-mover-password-secret-ref <my-optional-data-mover-secret> \
-n trident-protect
```

Soutenu `providerConfig.s3` options de configuration

Consultez le tableau suivant pour connaître les options de configuration du fournisseur S3 :

Paramètre	Description	Valeur par défaut	Exemple
`providerConfig.s3.skipCertValidation`	Désactiver la vérification des certificats SSL/TLS.	faux	"vrai", "faux"
`providerConfig.s3.secure`	Activez la communication HTTPS sécurisée avec le point de terminaison S3.	vrai	"vrai", "faux"
`providerConfig.s3.proxyURL`	Spécifiez l'URL du serveur proxy utilisé pour se connecter à S3.	Aucune	http://proxy.example.com:8080
`providerConfig.s3.rootCA`	Fournissez un certificat d'autorité de certification racine personnalisé pour la vérification SSL/TLS.	Aucune	"CN=MyCustomCA"
`providerConfig.s3.useIAM`	Activez l'authentification IAM pour accéder aux compartiments S3. Applicable à l'identité du pod EKS.	faux	vrai, faux

Afficher les informations AppVault

Vous pouvez utiliser le plugin CLI Trident Protect pour consulter les informations relatives aux objets AppVault que vous avez créés sur le cluster.

Étapes

1. Afficher le contenu d'un objet AppVault :

```
tridentctl-protect get appvaultcontent gcp-vault \
--show-resources all \
-n trident-protect
```

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 du cluster dans la première colonne du tableau n'est disponible que si un nom de cluster a été spécifié dans l'installation Helm de Trident Protect. Par exemple: `--set clusterName=production1` .

Supprimer un AppVault

Vous pouvez supprimer un objet AppVault à tout moment.

REMARQUE	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 CRS de snapshot et de sauvegarde utilisés par l'AppVault que vous souhaitez supprimer.

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
```

Supprimer un AppVault à l'aide de l'interface de ligne de commande Trident Protect

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 de gestion avec Trident Protect

Vous pouvez définir une application que vous souhaitez gérer avec Trident Protect en créant une demande de changement d'application et une demande de changement AppVault associée.

Créez une CR AppVault

Vous devez créer une ressource personnalisée AppVault qui sera utilisée lors des opérations de protection des données sur l'application, et cette ressource personnalisée AppVault doit résider sur le cluster où Trident Protect est installé. La demande de changement (CR) AppVault est spécifique à votre environnement ; pour des exemples de CR AppVault, reportez-vous à"Ressources personnalisées AppVault."

Définir une application

Vous devez définir chaque application que vous souhaitez gérer avec Trident Protect. Vous pouvez définir une application de gestion soit en créant manuellement une demande de changement d'application, soit en utilisant l'interface de ligne de commande (CLI) de Trident Protect.

REMARQUE	Pour définir une application à portée de machine virtuelle qui protège des machines virtuelles KubeVirt spécifiques par leur nom plutôt que des espaces de noms entiers, utilisez le champ `spec.includedVirtualMachines` dans la ressource personnalisée (Application CR) ou le paramètre `--virtual-machines` dans l'interface de ligne de commande (CLI). Pour plus de détails, la référence des champs et des exemples YAML, consultez "Protégez les machines virtuelles KubeVirt avec Trident Protect".

Ajouter une application à l'aide d'une demande de modification

Étapes

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 l'espace de noms et le sélecteur d'étiquettes pour spécifier les espaces de noms et les ressources utilisés par l'application. L'espace de nom de l'application doit faire partie de cette liste. Le sélecteur d'étiquettes est facultatif et peut être utilisé pour filtrer les ressources dans chaque espace de noms spécifié.
      - **spec.includedClusterScopedResources**: (*Optional*) utilisez cet attribut pour spécifier les ressources cluster-scoped à inclure dans la définition de l'application. Cet attribut vous permet de sélectionner ces ressources en fonction de leur groupe, de leur version, de leur type et de leurs étiquettes.
        - **GroupVersionKind**: (*required*) Spécifie le groupe d'API, la version et le type de la ressource cluster-scoped.
        - **LabelSelector**: (*Optional*) filtre les ressources du cluster-scoped en fonction de leurs étiquettes.
      - **metadata.annotations.protect.trident.netapp.io/skip-vm-freeze**: (*Optional*) Cette annotation s'applique uniquement aux applications définies à partir de machines virtuelles, comme dans les environnements KubeVirt, où les gels du système de fichiers se produisent avant les instantanés. Indiquez si cette application peut écrire sur le système de fichiers lors d'un instantané. Si cette option est activée (true), l'application ignore le paramètre global et peut écrire sur le système de fichiers pendant la création d'un instantané. Si cette option est désactivée, l'application ignore le paramètre global et le système de fichiers est figé lors de la création d'un instantané. Si cette annotation est spécifiée mais que l'application ne comporte aucune machine virtuelle dans sa définition, elle est ignorée. Sauf indication contraire, la demande suit la procédure"Paramètre de gel global Trident Protect" .

Si vous devez appliquer cette annotation après la création d'une application, vous pouvez utiliser la commande suivante :

```
kubectl annotate application -n <application CR namespace> <application CR name> protect.trident.netapp.io/skip-vm-freeze="true"
```

+
Exemple YAML :

+

```
apiVersion: protect.trident.netapp.io/v1
kind: Application
metadata:
  annotations:
    protect.trident.netapp.io/skip-vm-freeze: "false"
  name: my-app-name
  namespace: my-app-namespace
spec:
  includedNamespaces:
    - namespace: namespace-1
      labelSelector:
        matchLabels:
          app: example-app
    - namespace: namespace-2
      labelSelector:
        matchLabels:
          app: another-example-app
  includedClusterScopedResources:
    - groupVersionKind:
        group: rbac.authorization.k8s.io
        kind: ClusterRole
        version: v1
      labelSelector:
        matchLabels:
          mylabel: test
```

1. (*Facultatif*) Si nécessaire, vous pouvez ajouter un filtrage des ressources à la même CR pour inclure ou exclure des ressources spécifiques :

   - **Exemple de filtre générique :**

     - **ResourceFilter.resourceSelectionCriteria**: (Requis pour le filtrage) utiliser `Include` ou `Exclude` inclure ou exclure une ressource définie dans resourceMatchers. Ajoutez les paramètres resourceMatchers suivants pour définir les ressources à inclure ou à exclure :

       - **ResourceFilter.resourceMatchers** : un tableau d'objets resourceMatcher. Si vous définissez plusieurs éléments dans ce tableau, ils correspondent en tant qu'opération OU et les champs de chaque élément (groupe, type, version) correspondent en tant qu'opération ET.
       - **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"`.

REMARQUE	Quand les deux `resourceFilter` et `labelSelector` sont utilisés, `resourceFilter` court d'abord, puis `labelSelector` est appliqué aux ressources résultantes.

Par exemple :

```
spec:
  resourceFilter:
    resourceSelectionCriteria: "Include"
    resourceMatchers:
      - group: my-resource-group-1
        kind: my-resource-kind-1
        version: my-resource-version-1
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
      - group: my-resource-group-2
        kind: my-resource-kind-2
        version: my-resource-version-2
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
```

◦ **Exemple de filtre uniquement PVC :**

Pour définir une application utilisant uniquement des PVC, vous devez également inclure `PersistentVolume` et `VolumeSnapshotClass` dans le filtre de ressources. Les opérations de snapshot et de sauvegarde dépendent de `PersistentVolume` (le volume à portée du cluster lié à chaque PVC) et de `VolumeSnapshotClass` (le pilote de snapshot), et échoueront sans eux. Par exemple :

```
apiVersion: protect.trident.netapp.io/v1
kind: Application
metadata:
  name: my-pvc-app
  namespace: my-app-namespace
spec:
  includedNamespaces:
  - namespace: my-app-namespace
  resourceFilter:
    resourceMatchers:
    - kind: PersistentVolumeClaim
      version: v1
    - kind: PersistentVolume
      version: v1
    - kind: VolumeSnapshotClass
      version: v1
    resourceSelectionCriteria: Include
```

2. Une fois que vous avez créé la demande de modification de l'application pour l'adapter à votre environnement, appliquez la demande de modification. Par exemple :

```
kubectl apply -f maria-app.yaml
```

Étapes

1. Créez et appliquez la définition de l'application à l'aide de l'un des exemples suivants, 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 de l'application à l'aide de listes séparées par des virgules avec les arguments présentés dans les exemples.

Vous pouvez, si vous le souhaitez, utiliser une annotation lors de la création d'une application pour spécifier si celle-ci peut écrire sur le système de fichiers pendant la prise d'un instantané. Ceci ne s'applique qu'aux applications définies à partir de machines virtuelles, comme dans les environnements KubeVirt, où des blocages du système de fichiers se produisent avant la création des instantanés. Si vous définissez l'annotation sur `true` L'application ignore le paramètre global et peut écrire sur le système de fichiers pendant la création d'un instantané. Si vous le configurez à `false` L'application ignore alors le paramètre global et le système de fichiers est figé lors de la prise d'un instantané. Si vous utilisez l'annotation mais que l'application ne comporte aucune machine virtuelle dans sa définition, l'annotation est ignorée. Si vous n'utilisez pas l'annotation, l'application suit le comportement attendu."Paramètre de gel global Trident Protect" .

+ Pour spécifier l'annotation lorsque vous utilisez l'interface de ligne de commande pour créer une application, vous pouvez utiliser le `--annotation` indicateur.

+ * Créez l'application et utilisez le paramètre global pour le comportement de gel du système de fichiers :

+

```
tridentctl-protect create application <my_new_app_cr_name> --namespaces
<namespaces_to_include> --csr <cluster_scoped_resources_to_include>
--namespace <my-app-namespace>
```

+ * Créez l'application et configurez le paramètre local de l'application pour le comportement de gel du système de fichiers :

+

```
tridentctl-protect create application <my_new_app_cr_name> --namespaces
<namespaces_to_include> --csr <cluster_scoped_resources_to_include>
--namespace <my-app-namespace> --annotation
protect.trident.netapp.io/skip-vm-freeze=<"true"|"false">
```

Vous pouvez utiliser `--resource-filter-include` et `--resource-filter-exclude` pour inclure ou exclure des ressources en fonction de `resourceSelectionCriteria` tels que le groupe, le type, la version, les étiquettes, les noms et les espaces de noms, comme illustré dans l'exemple suivant :

+

```
tridentctl-protect create application <my_new_app_cr_name> --namespaces
<namespaces_to_include> --csr <cluster_scoped_resources_to_include>
--namespace <my-app-namespace> --resource-filter-include
'[{"Group":"apps","Kind":"Deployment","Version":"v1","Names":["my-
deployment"],"Namespaces":["my-namespace"],"LabelSelectors":["app=my-
app"]}]'
```

Pour définir une application utilisant uniquement des PVC, vous devez également inclure `PersistentVolume` et `VolumeSnapshotClass` dans le filtre de ressources. Les opérations de snapshot et d'archivage des sauvegardes dépendent de `PersistentVolume` (le volume à l'échelle du cluster lié à chaque PVC) et de `VolumeSnapshotClass` (le pilote de snapshot), et échoueront sans eux. Par exemple :

+

```
tridentctl-protect create app my-pvc-app --namespaces <my-app-
namespace> --resource-filter-include
'[{"Kind":"PersistentVolumeClaim","Version":"v1"},{"Kind":"PersistentVo
lume","Version":"v1"},{"Kind":"VolumeSnapshotClass","Version":"v1"}]'
-n <my-app-namespace>
```

Protégez les applications à l’aide de Trident Protect

Vous pouvez protéger toutes les applications gérées par Trident Protect en prenant des instantanés et des sauvegardes à l’aide d’une politique de protection automatisée ou de manière ponctuelle.

REMARQUE	Vous pouvez configurer Trident Protect pour geler et dégeler les systèmes de fichiers pendant les opérations de protection des données. "Apprenez-en davantage sur la configuration du gel du système de fichiers avec Trident Protect.".

Créer un snapshot à la demande

Vous pouvez créer un snapshot à la demande à tout moment.

REMARQUE	Les ressources Cluster-scoped sont incluses dans une sauvegarde, un snapshot ou un clone s’ils sont explicitement référencés dans la définition d’application ou s’ils ont des références à l’un des namespaces d’application.

Créer un instantané à l'aide d'une CR

Étapes

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

Étapes

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> -n
<application_namespace>
```

Créez une sauvegarde à la demande

Vous pouvez sauvegarder une application à tout moment.

REMARQUE	Les ressources Cluster-scoped sont incluses dans une sauvegarde, un snapshot ou un clone s'ils sont explicitement référencés dans la définition d'application ou s'ils ont des références à l'un des namespaces d'application.

Avant de commencer

Assurez-vous que l'expiration du jeton de session AWS suffit pour toutes les opérations de sauvegarde s3 à long terme. Si le jeton expire pendant l'opération de sauvegarde, l'opération peut échouer.

- Pour plus d'informations sur la vérification de l'expiration du jeton de session en cours, reportez-vous "Documentation de l'API AWS" au.
- Pour plus d'informations sur les identifiants avec les ressources AWS, consultez le "Documentation AWS IAM".

Créez une sauvegarde à l'aide d'une CR

Étapes

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.

     Exemple YAML :

```
---
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

Étapes

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> --data-mover <Kopia_or_Restic> -n <application_namespace>
```

Vous pouvez éventuellement utiliser `--full-backup` l'indicateur pour spécifier si une sauvegarde doit être non incrémentielle. Par défaut, toutes les sauvegardes sont incrémentielles. Lorsque cette option est utilisée, la sauvegarde devient non incrémentielle. Il est recommandé d'effectuer régulièrement une sauvegarde complète, puis d'effectuer des sauvegardes incrémentielles entre deux sauvegardes complètes afin de réduire le risque associé aux restaurations.

Annotations de sauvegarde prises en charge

Le tableau suivant décrit les annotations que vous pouvez utiliser lors de la création d'une demande de modification de sauvegarde :

Annotation	Type	Description	Valeur par défaut
protect.trident.netapp.io/full-backup	chaîne	Indique si la sauvegarde doit être non incrémentale. Réglé sur `true` créer une sauvegarde non incrémentale. Il est recommandé d'effectuer périodiquement une sauvegarde complète, puis d'effectuer des sauvegardes incrémentales entre les sauvegardes complètes afin de minimiser les risques liés aux restaurations.	« faux »
protect.trident.netapp.io/snapshot-completion-timeout	chaîne	Durée maximale autorisée pour l'exécution de l'opération globale de capture d'instantané.	"60m"
protect.trident.netapp.io/volume-snapshots-ready-to-use-timeout	chaîne	Le délai maximal autorisé pour que les instantanés de volume atteignent l'état prêt à l'emploi.	"30m"
protect.trident.netapp.io/volume-snapshots-created-timeout	chaîne	Durée maximale autorisée pour la création d'instantanés de volume.	"5m"
protect.trident.netapp.io/pvc-bind-timeout-sec	chaîne	Délai maximal (en secondes) d'attente pour que les PersistentVolumeClaims (PVC) nouvellement créées atteignent le `Bound` phase précédant l'échec des opérations.	"1200" (20 minutes)

Créez un calendrier de protection des données

Une politique de protection protège une application en créant des instantanés, des sauvegardes ou les deux selon un calendrier défini. Vous pouvez choisir de créer des instantanés 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. Vous pouvez planifier une sauvegarde complète non incrémentielle en utilisant l'annotation full-backup-rule. Par défaut, toutes les sauvegardes sont incrémentielles. L'exécution périodique d'une sauvegarde complète,

ainsi que de sauvegardes incrémentielles entre les deux, permet de réduire le risque associé aux restaurations.

REMARQUE

- Vous pouvez créer des planifications pour les instantanés uniquement en définissant `backupRetention` à zéro et `snapshotRetention` à une valeur supérieure à zéro. Paramètre `snapshotRetention` à zéro signifie que toutes les sauvegardes planifiées créeront toujours des instantanés, mais ceux-ci sont temporaires et seront supprimés immédiatement après la fin de la sauvegarde.
- Les ressources Cluster-scoped sont incluses dans une sauvegarde, un snapshot ou un clone s'ils sont explicitement référencés dans la définition d'application ou s'ils ont des références à l'un des namespaces d'application.

Créer un programme à l'aide d'une demande de modification

Étapes

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**: (*Obligatoire*) Le nombre de sauvegardes à conserver. Zéro indique qu'aucune sauvegarde ne doit être créée (instantanés uniquement).
   - **spec.backupReclaimPolicy**: (*Optionnel*) Détermine ce qui arrive à une sauvegarde si la ressource CR de sauvegarde est supprimée pendant sa période de rétention. Après la période de conservation, les sauvegardes sont toujours supprimées. Valeurs possibles (sensible à la casse) :
     - `Retain` (par défaut)
     - `Delete`
   - **spec.snapshotRetention**: (*Obligatoire*) Le nombre de snapshots à conserver. La valeur zéro indique qu'aucun instantané ne doit être créé.
   - **spec.snapshotReclaimPolicy**: (*Optionnel*) Détermine ce qui arrive à un instantané si la ressource personnalisée (CR) de l'instantané est supprimée pendant sa période de rétention. Après la période de conservation, les instantanés sont toujours supprimés. Valeurs possibles (sensible à la casse) :
     - `Retain`
     - `Delete`(défaut)
   - **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`)
     - `Custom`
   - **spec.dayOfMonth**: (*Facultatif*) Le jour du mois (1 - 31) auquel la planification doit s'exécuter. Ce champ est obligatoire si la granularité est définie sur `Monthly` . La valeur doit être fournie sous forme de chaîne.

- **spec.dayOfWeek**: (*Facultatif*) Le jour de la semaine (0 - 7) pendant lequel la planification doit s'exécuter. Les valeurs de 0 ou 7 indiquent dimanche. Ce champ est obligatoire si la granularité est définie sur `Weekly` . La valeur doit être fournie sous forme de chaîne.
- **spec.hour**: (*Facultatif*) L'heure de la journée (0 - 23) à laquelle le programme doit s'exécuter. Ce champ est obligatoire si la granularité est définie sur `Daily` , `Weekly` , ou `Monthly` . La valeur doit être fournie sous forme de chaîne.
- **spec.minute**: (*Facultatif*) La minute de l'heure (0 - 59) à laquelle la planification doit s'exécuter. Ce champ est obligatoire si la granularité est définie sur `Hourly` , `Daily` , `Weekly` , ou `Monthly` . La valeur doit être fournie sous forme de chaîne.
- **spec.runImmediately** : (*Optionnel*) Définissez sur `true` pour déclencher une exécution de base unique et immédiate (sauvegarde et/ou snapshot selon les paramètres de rétention) lors de la création de la planification. La valeur par défaut est `false`. Cela ne modifie pas la récurrence ultérieure.

  Exemple de YAML pour la planification de sauvegarde et de snapshot :

```
---
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: Daily
  hour: "0"
  minute: "0"
```

  Exemple de YAML pour la planification des instantanés uniquement :

```
---
apiVersion: protect.trident.netapp.io/v1
kind: Schedule
metadata:
  namespace: my-app-namespace
  name: my-snapshot-schedule
spec:
  applicationRef: my-application
  appVaultRef: appvault-name
  backupRetention: "0"
  snapshotRetention: "15"
  granularity: Daily
  hour: "2"
  minute: "0"
```

Exemple YAML pour une planification avec exécution immédiate :

```
---
apiVersion: protect.trident.netapp.io/v1
kind: Schedule
metadata:
  namespace: my-app-namespace
  name: my-daily-schedule-run-immediately
spec:
  applicationRef: my-application
  appVaultRef: appvault-name
  backupRetention: "7"
  snapshotRetention: "7"
  granularity: Daily
  hour: "3"
  minute: "0"
  runImmediately: true
```

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

Étapes

1. Créez le planning de protection en remplaçant les valeurs entre parenthèses par les informations de votre environnement. Par exemple :

REMARQUE 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> \
  --backup-reclaim-policy <Retain|Delete (default Retain)> \
  --data-mover <Kopia_or_Restic> \
  --day-of-month <day_of_month_to_run_schedule> \
  --day-of-week <day_of_week_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> \
  --snapshot-reclaim-policy <Retain|Delete (default Delete)> \
  --full-backup-rule <string> \
  --run-immediately <true|false> \
  -n <application_namespace>
```

Les options suivantes vous offrent un contrôle supplémentaire sur votre emploi du temps :

- **Planification complète des sauvegardes** : Utilisez la `--full-backup-rule` Option permettant de planifier des sauvegardes complètes non incrémentales. Ce drapeau ne fonctionne qu'avec `--granularity Daily`. Valeurs possibles :
  - `Always`: Créez une sauvegarde complète chaque jour.
  - Jours de la semaine spécifiques : Indiquez un ou plusieurs jours séparés par des virgules (par exemple, `"Monday,Thursday"`). Valeurs valides : lundi, mardi, mercredi, jeudi, vendredi, samedi, dimanche.

    REMARQUE Le `--full-backup-rule` Cette fonction ne fonctionne pas avec une granularité horaire, hebdomadaire ou mensuelle.

- **Protection immédiate de la base de référence** : Utilisez `--run-immediately true` pour créer une sauvegarde ou un instantané initial immédiatement lors de la création de la planification, plutôt que d'attendre la première exécution planifiée. La valeur par défaut est `false`.
- **Planifications instantanées uniquement** : Définir `--backup-retention 0` et spécifiez une valeur supérieure à zéro pour `--snapshot-retention`.

Annotations de planification prises en charge

Le tableau suivant décrit les annotations que vous pouvez utiliser lors de la création d'une demande de changement (CR) de planification :

Annotation	Type	Description	Valeur par défaut
protect.trident.netapp.io/full-backup-rule	chaîne	Spécifie la règle de planification des sauvegardes complètes. Vous pouvez le paramétrer pour `Always` pour une sauvegarde complète et permanente ou personnalisez-la en fonction de vos besoins. Par exemple, si vous choisissez une granularité quotidienne, vous pouvez spécifier les jours de la semaine où la sauvegarde complète doit avoir lieu (par exemple, `"Monday,Thursday"`). Les jours de semaine valides sont : lundi, mardi, mercredi, jeudi, vendredi, samedi, dimanche. Notez que cette annotation ne peut être utilisée qu'avec les calendriers qui possèdent `granularity` défini à `Daily`.	Non configuré (toutes les sauvegardes sont incrémentales)
protect.trident.netapp.io/snapshot-completion-timeout	chaîne	Durée maximale autorisée pour l'exécution de l'opération globale de capture d'instantané.	"60m"
protect.trident.netapp.io/volume-snapshots-ready-to-use-timeout	chaîne	Le délai maximal autorisé pour que les instantanés de volume atteignent l'état prêt à l'emploi.	"30m"
protect.trident.netapp.io/volume-snapshots-created-timeout	chaîne	Durée maximale autorisée pour la création d'instantanés de volume.	"5m"
protect.trident.netapp.io/pvc-bind-timeout-sec	chaîne	Délai maximal (en secondes) d'attente pour que les PersistentVolumeClaims (PVC) nouvellement créées atteignent le `Bound` phase précédant l'échec des opérations.	"1200" (20 minutes)

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.

REMARQUE	Assurez-vous que la politique de récupération est définie sur `Delete` pour supprimer toutes les données de sauvegarde du stockage d'objets. Le paramètre par défaut de la politique est `Retain` pour éviter toute perte accidentelle de données. Si la politique n'est pas modifiée `Delete` , les données de sauvegarde resteront dans le stockage d'objets et nécessiteront une suppression manuelle.

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 une fonctionnalité de sauvegarde et de restauration économe en espace pour les backends de stockage qui utilisent la classe de stockage azure-netapp-files et qui ont été créés avant Trident 24.06. Cette fonctionnalité est compatible avec les volumes NFSv4 et ne consomme pas d'espace supplémentaire du 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 offrira des fonctionnalités de protection limitées 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 de disposer d'un espace suffisant dans le pool de capacité pour créer un volume temporaire de la taille du volume sauvegardé.

Résultat

L'application est prête pour la sauvegarde et la restauration à l'aide de Trident Protect. Chaque PVC peut également être utilisé par d'autres applications pour les sauvegardes et les restaurations.

Restauration des applications

Restaurez les applications à l'aide de Trident Protect

Vous pouvez utiliser Trident Protect pour restaurer votre application à partir d'un instantané ou d'une sauvegarde. La restauration à partir d'un instantané existant sera plus rapide lors de la restauration de l'application sur le même cluster.

REMARQUE

- 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.
- La restauration à partir d'une sauvegarde vers un espace de noms différent ou vers l'espace de noms d'origine est prise en charge pour les volumes qtree. En revanche, la restauration à partir d'un snapshot vers un espace de noms différent ou vers l'espace de noms d'origine n'est pas prise en charge pour les volumes qtree.
- Vous pouvez utiliser des paramètres avancés pour personnaliser les opérations de restauration. Pour en savoir plus, consultez "Utilisez les paramètres de restauration avancés de Trident Protect".
- Pour restaurer des machines virtuelles (VM) KubeVirt spécifiques à partir d'une archive sans restaurer toutes les ressources de l'espace de noms, utilisez le champ `includedVirtualMachines` sur une CR de restauration. Pour les CR prises en charge, la référence des champs et des exemples YAML, consultez "Protégez les machines virtuelles KubeVirt avec Trident Protect".

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 ressource personnalisée BackupRestore, Trident Protect restaure l'application dans un nouvel espace de noms et crée une ressource personnalisée d'application pour l'application restaurée. Pour protéger l'application restaurée, créez des sauvegardes ou des instantanés à la demande, ou établissez un calendrier de protection.

REMARQUE

- 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.
- Lorsque vous utilisez une modification de configuration (CR) pour restaurer un nouvel espace de noms, vous devez créer manuellement l'espace de noms de destination avant d'appliquer la CR. Trident Protect crée automatiquement des espaces de noms uniquement lors de l'utilisation de l'interface de ligne de commande (CLI).

Avant de commencer

Assurez-vous que l'expiration du jeton de session AWS suffit pour toutes les opérations de restauration s3 à long terme. Si le jeton expire pendant l'opération de restauration, l'opération peut échouer.

- Pour plus d'informations sur la vérification de l'expiration du jeton de session en cours, reportez-vous "Documentation de l'API AWS" au.
- Pour plus d'informations sur les identifiants avec les ressources AWS, consultez le "Documentation AWS IAM".

REMARQUE	Lorsque vous restaurez des sauvegardes à l'aide de Kopia comme outil de transfert de données, vous pouvez éventuellement spécifier des annotations dans le CR ou utiliser l'interface de ligne de commande pour contrôler le comportement du stockage temporaire utilisé par Kopia. Se référer à "Documentation Kopia" pour plus d'informations sur les options que vous pouvez configurer. Utilisez le `tridentctl-protect create --help` commande pour plus d'informations sur la spécification des annotations avec l'interface de ligne de commande Trident Protect.

Utiliser une CR

Étapes

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.destinationApplicationName** : (*Facultatif*) Le nom de l'application restaurée. Si ce nom est fourni, l'application restaurée utilise ce nom. Si ce nom n'est pas fourni, l'application restaurée utilise le nom de l'application source.
   - **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: BackupRestore
     metadata:
       name: my-cr-name
       namespace: my-destination-namespace
     spec:
       appArchivePath: my-backup-path
       appVaultRef: appvault-name
       destinationApplicationName: my-new-app-name
       namespaceMapping: [{"source": "my-source-namespace",
     "destination": "my-destination-namespace"}]
     ```

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 :

REMARQUE	Trident Protect sélectionne automatiquement certaines ressources en raison de leur relation avec les ressources que vous sélectionnez. Par exemple, si vous sélectionnez une ressource de revendication de volume persistant et qu'elle possède un pod associé, Trident Protect restaurera également le pod associé.

   - **ResourceFilter.resourceSelectionCriteria**: (Requis pour le filtrage) utiliser `Include` ou

`Exclude` inclure ou exclure une ressource définie dans resourceMatchers. Ajoutez les paramètres resourceMatchers suivants pour définir les ressources à inclure ou à exclure :

- **ResourceFilter.resourceMatchers** : un tableau d'objets resourceMatcher. Si vous définissez plusieurs éléments dans ce tableau, ils correspondent en tant qu'opération OU et les champs de chaque élément (groupe, type, version) correspondent en tant qu'opération ET.
  - **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-1
        kind: my-resource-kind-1
        version: my-resource-version-1
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
      - group: my-resource-group-2
        kind: my-resource-kind-2
        version: my-resource-version-2
        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

Étapes

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> \
--destination-app-name<custom_app_name>\
-n <application_namespace>
```

Restaurer à partir d'une sauvegarde vers l'espace de noms d'origine

Vous pouvez restaurer une sauvegarde dans l'espace de noms d'origine à tout moment. Lorsque vous effectuez une restauration sur place, Trident Protect gère automatiquement les planifications de protection et les opérations en cours afin d'éviter les points de récupération invalides :

- Tous les plans de protection activés pour l'application sont désactivés avant le début de la restauration. Cela empêche les sauvegardes planifiées ou les instantanés de s'exécuter pendant la restauration des ressources de l'application.
- Après la restauration réussie, seules les planifications qui étaient activées avant la restauration sont réactivées. Les planifications qui étaient déjà désactivées restent désactivées.
- Toute opération de sauvegarde ou de capture instantanée en cours est annulée avant le début de la restauration. Si une opération n'est pas annulée dans les 5 minutes, la restauration se poursuit et un avertissement est consigné dans le statut du CR de restauration.

Avant de commencer

Assurez-vous que l'expiration du jeton de session AWS suffit pour toutes les opérations de restauration s3 à long terme. Si le jeton expire pendant l'opération de restauration, l'opération peut échouer.

- Pour plus d'informations sur la vérification de l'expiration du jeton de session en cours, reportez-vous "Documentation de l'API AWS" au.
- Pour plus d'informations sur les identifiants avec les ressources AWS, consultez le "Documentation AWS IAM".

REMARQUE	Lorsque vous restaurez des sauvegardes à l'aide de Kopia comme outil de transfert de données, vous pouvez éventuellement spécifier des annotations dans le CR ou utiliser l'interface de ligne de commande pour contrôler le comportement du stockage temporaire utilisé par Kopia. Se référer à "Documentation Kopia" pour plus d'informations sur les options que vous pouvez configurer. Utilisez le `tridentctl-protect create --help` commande pour plus d'informations sur la spécification des annotations avec l'interface de ligne de commande Trident Protect.

Utiliser une CR

Étapes

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 :

REMARQUE	Trident Protect sélectionne automatiquement certaines ressources en raison de leur relation avec les ressources que vous sélectionnez. Par exemple, si vous sélectionnez une ressource de revendication de volume persistant et qu'elle possède un pod associé, Trident Protect restaurera également le pod associé.

   - **ResourceFilter.resourceSelectionCriteria**: (Requis pour le filtrage) utiliser `Include` ou `Exclude` inclure ou exclure une ressource définie dans resourceMatchers. Ajoutez les paramètres resourceMatchers suivants pour définir les ressources à inclure ou à exclure :
     - **ResourceFilter.resourceMatchers** : un tableau d'objets resourceMatcher. Si vous définissez plusieurs éléments dans ce tableau, ils correspondent en tant qu'opération OU et les champs de chaque élément (groupe, type, version) correspondent en tant qu'opération ET.
       - **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-1
        kind: my-resource-kind-1
        version: my-resource-version-1
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
      - group: my-resource-group-2
        kind: my-resource-kind-2
        version: my-resource-version-2
        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

Étapes

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> \
-n <application_namespace>
```

Restauration à partir d'une sauvegarde sur un autre cluster

Vous pouvez restaurer une sauvegarde sur un autre cluster en cas de problème avec le cluster d'origine.

REMARQUE

- Lorsque vous restaurez des sauvegardes à l'aide de Kopia comme outil de transfert de données, vous pouvez éventuellement spécifier des annotations dans le CR ou utiliser l'interface de ligne de commande pour contrôler le comportement du stockage temporaire utilisé par Kopia. Se référer à "Documentation Kopia" pour plus d'informations sur les options que vous pouvez configurer. Utilisez le `tridentctl-protect create --help` commande pour plus d'informations sur la spécification des annotations avec l'interface de ligne de commande Trident Protect.
- Lorsque vous utilisez une modification de configuration (CR) pour restaurer un nouvel espace de noms, vous devez créer manuellement l'espace de noms de destination avant d'appliquer la CR. Trident Protect crée automatiquement des espaces de noms uniquement lors de l'utilisation de l'interface de ligne de commande (CLI).

Avant de commencer

Assurez-vous que les conditions préalables suivantes sont remplies :

- Le cluster de destination possède Trident Protect installé.
- Le cluster de destination a accès au chemin de compartiment du même AppVault que le cluster source, où la sauvegarde est stockée.
- Assurez-vous que votre environnement local peut se connecter au compartiment de stockage d'objets défini dans la ressource personnalisée AppVault lors de l'exécution. `tridentctl-protect get appvaultcontent` commande. Si des restrictions réseau empêchent l'accès, exécutez plutôt l'interface de ligne de commande Trident Protect depuis un pod sur le cluster de destination.
- Assurez-vous que l'expiration du jeton de session AWS suffit pour toutes les opérations de restauration à long terme. Si le jeton expire pendant l'opération de restauration, l'opération peut échouer.
  - Pour plus d'informations sur la vérification de l'expiration du jeton de session en cours, reportez-vous "Documentation de l'API AWS" au.
  - Pour plus d'informations sur les identifiants avec les ressources AWS, consultez le "Documentation de l'AWS".

Étapes

1. Vérifiez que la AppVault CR existe sur le cluster de destination à l'aide du plugin CLI Trident Protect :

```
tridentctl-protect get appvault --context <destination_cluster_name>
```

REMARQUE

Si la AppVault CR n'existe pas sur le cluster de destination, créez-la en suivant les étapes dans "Utilisez les objets Trident Protect AppVault pour gérer les compartiments.".

2. Consultez le contenu de la sauvegarde disponible de AppVault sur le cluster de destination, et notez `appArchivePath` de la sauvegarde que vous souhaitez restaurer :

```
tridentctl-protect get appvaultcontent <appvault_name> \
--show-resources backup \
--show-paths \
--context <destination_cluster_name>
```

L'exécution de cette commande affiche les sauvegardes disponibles dans le AppVault, y compris leurs clusters d'origine, les noms d'applications correspondants, les horodatages et les chemins d'archivage.

Exemple de sortie :

```
+-------------+-----------+--------+-----------------
+--------------------------+-------------+
|   CLUSTER   |    APP    |  TYPE  |      NAME       |         TIMESTAMP
|    PATH     |
+-------------+-----------+--------+-----------------
+--------------------------+-------------+
| production1 | wordpress | backup | wordpress-bkup-1| 2024-10-30
08:37:40 (UTC)| backuppath1 |
| production1 | wordpress | backup | wordpress-bkup-2| 2024-10-30
08:37:40 (UTC)| backuppath2 |
+-------------+-----------+--------+-----------------
+--------------------------+-------------+
```

3. Restaurez l'application sur le cluster de destination à l'aide du nom AppVault et du chemin d'archivage :

REMARQUE	Lors de l'utilisation d'une CR, assurez-vous que l'espace de noms destiné à la restauration de l'application existe sur le cluster de destination.

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.appVaultRef**: (*required*) Nom de l'AppVault où sont stockés le contenu de la sauvegarde.
   - **spec.appArchivePath** : (Obligatoire) Le chemin à l'intérieur de AppVault où le contenu de la sauvegarde est stocké. Utilisez la commande de l'étape 2 pour afficher le contenu de la sauvegarde et trouver `appArchivePath` la sauvegarde que vous souhaitez restaurer.
   - **spec.destinationApplicationName** : (*Facultatif*) Le nom de l'application restaurée. Si ce nom est fourni, l'application restaurée utilise ce nom. Si ce nom n'est pas fourni, l'application restaurée utilise le nom de l'application source.
   - **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.

     Par exemple :

```
apiVersion: protect.trident.netapp.io/v1
kind: BackupRestore
metadata:
  name: my-cr-name
  namespace: my-destination-namespace
spec:
  appVaultRef: appvault-name
  appArchivePath: my-backup-path
  destinationApplicationName: my-new-app-name
  namespaceMapping: [{"source": "my-source-namespace", "
destination": "my-destination-namespace"}]
```

3. 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. Utilisez la commande suivante pour restaurer l'application, en remplaçant les valeurs entre parenthèses par les informations de votre environnement. L'argument namespace-mapping utilise des espaces de noms séparés par deux points pour mapper les espaces de noms source aux espaces de noms de destination corrects au format source1:dest1,source2:dest2. Par exemple :

```
tridentctl-protect create backuprestore <restore_name> \
--namespace-mapping <source_to_destination_namespace_mapping> \
--appvault <appvault_name> \
--path <backup_path> \
--destination-app-name <custom_app_name> \
--context <destination_cluster_name> \
-n <application_namespace>
```

Restauration d'un snapshot vers un autre espace de noms

Vous pouvez restaurer des données à partir d'un instantané à l'aide d'un fichier de ressources personnalisé (CR), soit dans un espace de noms différent, soit dans l'espace de noms source d'origine. Lorsque vous restaurez un instantané dans un espace de noms différent à l'aide d'une ressource personnalisée SnapshotRestore, Trident Protect restaure l'application dans un nouvel espace de noms et crée une ressource personnalisée d'application pour l'application restaurée. Pour protéger l'application restaurée, créez des sauvegardes ou des instantanés à la demande, ou établissez un calendrier de protection.

REMARQUE

- SnapshotRestore prend en charge le `spec.storageClassMapping` attribut, mais uniquement lorsque les classes de stockage source et de destination utilisent le même backend de stockage. Si vous tentez de restaurer un `StorageClass` qui utilise un backend de stockage différent, l'opération de restauration échouera.
- Lorsque vous utilisez une modification de configuration (CR) pour restaurer un nouvel espace de noms, vous devez créer manuellement l'espace de noms de destination avant d'appliquer la CR. Trident Protect crée automatiquement des espaces de noms uniquement lors de l'utilisation de l'interface de ligne de commande (CLI).

Avant de commencer

Assurez-vous que l'expiration du jeton de session AWS suffit pour toutes les opérations de restauration s3 à long terme. Si le jeton expire pendant l'opération de restauration, l'opération peut échouer.

- Pour plus d'informations sur la vérification de l'expiration du jeton de session en cours, reportez-vous "Documentation de l'API AWS" au.
- Pour plus d'informations sur les identifiants avec les ressources AWS, consultez le "Documentation AWS IAM".

Utiliser une CR

Étapes

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.destinationApplicationName** : (*Facultatif*) Le nom de l'application restaurée. Si ce nom est fourni, l'application restaurée utilise ce nom. Si ce nom n'est pas fourni, l'application restaurée utilise le nom de l'application source.
   - **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: my-destination-namespace
     spec:
       appVaultRef: appvault-name
       appArchivePath: my-snapshot-path
       namespaceMapping: [{"source": "my-source-namespace",
     "destination": "my-destination-namespace"}]
     ```

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 :

   REMARQUE Trident Protect sélectionne automatiquement certaines ressources en raison de leur relation avec les ressources que vous sélectionnez. Par exemple, si vous sélectionnez une ressource de revendication de volume persistant et qu'elle possède un pod associé, Trident Protect restaurera également le pod associé.

   - **ResourceFilter.resourceSelectionCriteria**: (Requis pour le filtrage) utiliser `Include` ou

`Exclude` inclure ou exclure une ressource définie dans resourceMatchers. Ajoutez les paramètres resourceMatchers suivants pour définir les ressources à inclure ou à exclure :

- **ResourceFilter.resourceMatchers** : un tableau d'objets resourceMatcher. Si vous définissez plusieurs éléments dans ce tableau, ils correspondent en tant qu'opération OU et les champs de chaque élément (groupe, type, version) correspondent en tant qu'opération ET.
  - **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-1
        kind: my-resource-kind-1
        version: my-resource-version-1
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
      - group: my-resource-group-2
        kind: my-resource-kind-2
        version: my-resource-version-2
        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

Étapes

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> \
--destination-app-name <custom_app_name> \
-n <application_namespace>
```

Restaurer à partir d'un snapshot vers l'espace de noms d'origine

Vous pouvez restaurer un instantané dans l'espace de noms d'origine à tout moment. Lorsque vous effectuez une restauration sur place, Trident Protect gère automatiquement les planifications de protection et les opérations en cours afin d'éviter les points de récupération invalides :

- Tous les plans de protection activés pour l'application sont désactivés avant le début de la restauration. Cela empêche les sauvegardes planifiées ou les instantanés de s'exécuter pendant la restauration des ressources de l'application.
- Après la restauration réussie, seules les planifications qui étaient activées avant la restauration sont réactivées. Les planifications qui étaient déjà désactivées restent désactivées.
- Toute opération de sauvegarde ou de capture instantanée en cours est annulée avant le début de la restauration. Si une opération n'est pas annulée dans les 5 minutes, la restauration se poursuit et un avertissement est consigné dans le statut du CR de restauration.

Avant de commencer

Assurez-vous que l'expiration du jeton de session AWS suffit pour toutes les opérations de restauration s3 à long terme. Si le jeton expire pendant l'opération de restauration, l'opération peut échouer.

- Pour plus d'informations sur la vérification de l'expiration du jeton de session en cours, reportez-vous "Documentation de l'API AWS" au.
- Pour plus d'informations sur les identifiants avec les ressources AWS, consultez le "Documentation AWS IAM".

Utiliser une CR

Étapes

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
```

+ . (*Facultatif*) Si vous devez sélectionner uniquement certaines ressources de l'application à restaurer, ajoutez un filtrage qui inclut ou exclut les ressources marquées d'étiquettes particulières :

+ REMARQUE : Trident Protect sélectionne automatiquement certaines ressources en fonction de leur relation avec les ressources que vous sélectionnez. Par exemple, si vous sélectionnez une ressource de revendication de volume persistant et qu'elle possède un pod associé, Trident Protect restaurera également le pod associé.

+ * **resourceFilter.resourceSelectionCriteria** : (Requis pour le filtrage) Utilisez `Include` ou `Exclude` pour inclure ou exclure une ressource définie dans resourceMatchers. Ajoutez les paramètres suivants resourceMatchers pour définir les ressources à inclure ou à exclure : **resourceFilter.resourceMatchers : Un tableau d'objets resourceMatcher. Si vous définissez plusieurs éléments dans ce tableau, ils correspondent à une opération OU, et les champs à l'intérieur de chaque élément (group, kind, version) correspondent à une opération ET.** * **resourceMatchers[].group** : (*Optionnel*) Groupe de la ressource à filtrer. *** resourceMatchers[].kind : (*Optionnel*) Type de la ressource à filtrer.** * **resourceMatchers[].version** : (*Optionnel*) Version de la ressource à filtrer. *** resourceMatchers[].names : (*Optionnel*) Noms dans le champ metadata.name de la ressource à filtrer.** * **resourceMatchers[].namespaces** : (*Optionnel*) Espaces de noms dans le champ metadata.name de la ressource à filtrer. *** resourceMatchers[].labelSelectors** : (*Optionnel*) Chaîne de sélection d'étiquette dans le champ metadata.name de la ressource Kubernetes tel que défini dans le

"Documentation Kubernetes". Par exemple : `"trident.netapp.io/os=linux"`.

+ Par exemple :

+

```
spec:
  resourceFilter:
    resourceSelectionCriteria: "Include"
    resourceMatchers:
      - group: my-resource-group-1
        kind: my-resource-kind-1
        version: my-resource-version-1
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
      - group: my-resource-group-2
        kind: my-resource-kind-2
        version: my-resource-version-2
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
```

+ . Après avoir rempli le `trident-protect-snapshot-ipr-cr.yaml` fichier avec les valeurs correctes, appliquez le CR :

+

```
kubectl apply -f trident-protect-snapshot-ipr-cr.yaml
```

Utiliser l'interface de ligne de commande

Étapes

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 <namespace/snapshot_to_restore> \
-n <application_namespace>
```

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}'
```

Utilisez les paramètres de restauration avancés de Trident Protect

Vous pouvez personnaliser les opérations de restauration à l'aide de paramètres avancés tels que les annotations, les paramètres d'espace de noms et les options de stockage pour répondre à vos besoins spécifiques.

Annotations et étiquettes de namespace pendant les opérations de restauration et de basculement

Lors des opérations de restauration et de basculement, les libellés et les annotations dans l'espace de noms de destination correspondent aux libellés et aux annotations dans l'espace de noms source. Des étiquettes ou des annotations provenant de l'espace de noms source qui n'existent pas dans l'espace de noms de destination sont ajoutées et toutes les étiquettes ou annotations qui existent déjà sont écrasées pour correspondre à la valeur de l'espace de noms source. Les libellés ou annotations qui existent uniquement dans l'espace de noms de destination restent inchangés.

REMARQUE	Si vous utilisez Red Hat OpenShift, il est important de noter le rôle essentiel des annotations d'espace de noms dans les environnements OpenShift. Les annotations d'espace de noms garantissent que les pods restaurés adhèrent aux autorisations et aux configurations de sécurité appropriées définies par les contraintes de contexte de sécurité (SCC) OpenShift et peuvent accéder aux volumes sans problèmes d'autorisation. Pour plus d'informations, reportez-vous à la"Documentation sur les contraintes de contexte de sécurité OpenShift" .

Vous pouvez empêcher l'écrasement d'annotations spécifiques dans l'espace de noms de destination en configurant la variable d'environnement Kubernetes `RESTORE_SKIP_NAMESPACE_ANNOTATIONS` avant d'effectuer l'opération de restauration ou de basculement. Par exemple :

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect \
  --set-string
restoreSkipNamespaceAnnotations="{<annotation_key_to_skip_1>,<annotation_k
ey_to_skip_2>}" \
  --reuse-values
```

REMARQUE	Lors d'une opération de restauration ou de basculement, toutes les annotations et étiquettes d'espace de noms spécifiées dans `restoreSkipNamespaceAnnotations` et `restoreSkipNamespaceLabels` sont exclues de l'opération de restauration ou de basculement. Assurez-vous que ces paramètres sont configurés lors de l'installation initiale de Helm. Pour en savoir plus, consultez "Configurer les paramètres supplémentaires du graphique de barre de Trident Protect".

Si vous avez installé l'application source à l'aide de Helm avec le `--create-namespace` Le drapeau, un traitement spécial est accordé au `name` Légende. Lors du processus de restauration ou de basculement, Trident Protect copie cette étiquette dans l'espace de noms de destination, mais met à jour la valeur avec la valeur de l'espace de noms de destination si la valeur de la source correspond à l'espace de noms source. Si cette valeur ne correspond pas à l'espace de noms source, elle est copiée dans l'espace de noms de destination sans modification.

Exemple

L'exemple suivant présente un espace de noms source et de destination, chacun avec des annotations et des libellés différents. Vous pouvez voir l'état de l'espace de noms de destination avant et après l'opération, ainsi que la manière dont les annotations et les étiquettes sont combinées ou écrasées dans l'espace de noms de destination.

Avant l'opération de restauration ou de basculement

Le tableau suivant illustre l'état de l'exemple d'espaces de noms source et de destination avant l'opération de restauration ou de basculement :

Espace de noms	Annotations	Étiquettes
Espace de noms ns-1 (source)	• annotation.one/key : « updatedvalue » • annotation.deux/touche : « vrai »	• environnement=production • conformité = hipaa • name=ns-1
Espace de noms ns-2 (destination)	• annotation.un/touche : « vrai » • annotation.trois/touche : « false »	• role=base de données

Après l'opération de restauration

Le tableau suivant illustre l'état de l'exemple d'espace de noms de destination après une opération de restauration ou de basculement. Certaines clés ont été ajoutées, d'autres ont été écrasées et le `name` libellé a été mis à jour pour correspondre à l'espace de noms de destination :

Espace de noms	Annotations	Étiquettes
Espace de noms ns-2 (destination)	• annotation.one/key : « updatedvalue » • annotation.deux/touche : « vrai » • annotation.trois/touche : « false »	• name=ns-2 • conformité = hipaa • environnement=production • role=base de données

Champs pris en charge

Cette section décrit les champs supplémentaires disponibles pour les opérations de restauration.

Mappage des classes de stockage

Le `spec.storageClassMapping` L'attribut définit un mappage d'une classe de stockage présente dans l'application source vers une nouvelle classe de stockage sur le cluster cible. Vous pouvez l'utiliser lors de la migration d'applications entre des clusters avec différentes classes de stockage ou lors du changement du backend de stockage pour les opérations BackupRestore.

Exemple:

```
storageClassMapping:
  - destination: "destinationStorageClass1"
    source: "sourceStorageClass1"
  - destination: "destinationStorageClass2"
    source: "sourceStorageClass2"
```

Annotations prises en charge

Cette section répertorie les annotations prises en charge pour configurer différents comportements du système. Si une annotation n'est pas explicitement définie par l'utilisateur, le système utilisera la valeur par défaut.

Annotation	Type	Description	Valeur par défaut
protect.trident.netapp.io/data-mover-timeout-sec	chaîne	Le temps maximal (en secondes) autorisé pour que le fonctionnement du moteur de transfert de données soit bloqué.	"300"
protect.trident.netapp.io/kopia-content-cache-size-limit-mb	chaîne	La limite de taille maximale (en mégaoctets) pour le cache de contenu Kopia.	"1000"
protect.trident.netapp.io/pvc-bind-timeout-sec	chaîne	Délai maximal (en secondes) d'attente pour que les PersistentVolumeClaims (PVC) nouvellement créées atteignent le `Bound` phase précédant l'échec des opérations. S'applique à tous les types de restauration CR (BackupRestore, BackupInplaceRestore, SnapshotRestore, SnapshotInplaceRestore). Utilisez une valeur plus élevée si votre système de stockage ou votre cluster nécessite souvent plus de temps.	"1200" (20 minutes)

Modifiez les ressources au fur et à mesure de leur restauration à l'aide de ressources personnalisées

Les transformations de ressources permettent de modifier une ressource pendant sa restauration. Ceci est utile lorsque la version restaurée doit différer de l'originale, par exemple pour mettre à jour un nom d'hôte dans un ConfigMap lors de la restauration

d'une application sur un cluster différent.

Comment fonctionne la modification des ressources

Le champ `transformations` dans `SnapshotRestore`, `BackupRestore`, `AppMirrorRelationship` et d'autres ressources de restauration vous permet de modifier les ressources Kubernetes pendant le processus de restauration. Ceci est utile lorsque l'application restaurée doit être configurée différemment de l'originale, par exemple en modifiant les noms d'hôtes, les URL de registre, les limites de ressources ou les variables d'environnement.

Les transformations de ressources utilisent "Correctif JSON RFC 6902" des opérations et "Pointeur JSON RFC 6901" des chemins pour cibler et modifier des champs spécifiques au sein des ressources Kubernetes.

La structure de base d'un objet de restauration contenant des transformations de ressources est présentée ci-dessous :

```
apiVersion: protect.trident.netapp.io/v1
kind: SnapshotRestore
metadata:
  name: my-restore
  namespace: target-namespace
spec:
  appVaultRef: my-vault
  appArchivePath: /path/to/snapshot
  namespaceMapping:
    - source: source-ns
      destination: target-ns
  transformations:
    - resource:
        kind: ConfigMap       # Required: resource kind
        group: ""             # Optional: API group (empty for core
resources)
        version: ""           # Optional: API version
        name: ""              # Optional: specific resource name
      operations:
        - op: replace         # Operation type
          path: "/data/key"   # JSON Pointer path
          value: "new-value"  # New value (for add/replace/test)
```

Ressources prises en charge

Vous pouvez utiliser les transformations de ressources avec les ressources qui correspondent aux critères suivants :

- **kind** (obligatoire) : Le type de ressource Kubernetes (par exemple, `ConfigMap`, `Deployment`, `Pod`)
- **groupe** (facultatif) : Le groupe d'API (par exemple, `apps`, `route.openshift.io`) — à omettre pour les ressources principales
- **version** (facultatif) : La version de l'API (par exemple, `v1`, `v1beta1`)

- **nom** (facultatif) : Appliquer uniquement à une ressource spécifique par nom

REMARQUE	La modification de PersistentVolumeClaims et des espaces de noms n'est actuellement pas prise en charge.

Opérations prises en charge

Vous pouvez utiliser les opérations suivantes pour modifier les ressources : `add`, `copy`, `move`, `remove`, `replace`, et `test`.

Ajouter une valeur à une ressource

Utilisez l' `add`opération pour ajouter un nouveau champ ou une nouvelle valeur au chemin spécifié. Vous pouvez ajouter des données à des objets ou à des tableaux.

Utiliser une CR

L'exemple suivant ajoute un sélecteur de nœud à une `Deployment` ressource :

```
transformations:
  - resource:
      kind: Deployment
    operations:
      - op: add
        path: "/spec/template/spec/nodeSelector"
        value:
          "topology.kubernetes.io/zone": "us-east-1a"
          disktype: "ssd"
```

Utiliser l'interface de ligne de commande

Utilisez la commande suivante pour exécuter cette transformation en ligne de commandes :

```
tridentctl-protect create snapshotrestore <restore_name> \
  --snapshot <namespace/snapshot-name> \
  --namespace-mapping <source-ns>:<dest-ns> \
  --transformation
'apps,v1,Deployment:add{"path":"/spec/template/spec/nodeSelector","valu
e":{"topology.kubernetes.io/zone":"us-east-1a","disktype":"ssd"}}'
```

Copier une valeur dans une ressource

Utilisez l' `copy`opération pour copier une valeur d'un chemin à un autre au sein de la même ressource. La source reste inchangée.

Utiliser une CR

L'exemple suivant duplique une clé de données pour un objet `ConfigMap` :

```
transformations:
  - resource:
      kind: ConfigMap
    operations:
      - op: copy
        from: "/data/source-key"
        path: "/data/backup-key"
```

Utiliser l'interface de ligne de commande

Utilisez la commande suivante pour exécuter cette transformation en ligne de commandes :

```
tridentctl-protect create snapshotrestore <restore_name> \
  --snapshot <namespace/snapshot-name> \
  --namespace-mapping <source-ns>:<dest-ns> \
  --transformation ',v1,ConfigMap:copy{"from":"/data/source-
key","path":"/data/backup-key"}'
```

Déplacer une valeur au sein d'une ressource

Utilisez l' `move`opération pour déplacer une valeur d'un chemin à un autre au sein de la même ressource. La source est supprimée et la valeur est placée à la destination.

Utiliser une CR

L'exemple suivant renomme une clé de données pour un objet `ConfigMap` :

```
transformations:
  - resource:
      kind: ConfigMap
    operations:
      - op: move
        from: "/data/OLD_KEY"
        path: "/data/NEW_KEY"
```

Utiliser l'interface de ligne de commande

Utilisez la commande suivante pour exécuter cette transformation en ligne de commandes :

```
tridentctl-protect create snapshotrestore <restore_name> \
  --snapshot <namespace/snapshot-name> \
  --namespace-mapping <source-ns>:<dest-ns> \
  --transformation
',v1,ConfigMap:move{"from":"/data/OLD_KEY","path":"/data/NEW_KEY"}'
```

Supprimer une valeur d'une ressource

Utilisez l' `remove`opération pour supprimer un champ ou une valeur au chemin spécifié.

Utiliser une CR

L'exemple suivant supprime une annotation d'une `ConfigMap` ressource :

```
transformations:
  - resource:
      kind: ConfigMap
    operations:
      - op: remove
        path: "/metadata/annotations/kubectl.kubernetes.io~1last-
applied-configuration"
```

REMARQUE Dans le chemin de l'exemple ci-dessus, `~1` est la séquence d'échappement du pointeur JSON pour `/`.

Utiliser l'interface de ligne de commande

Utilisez la commande suivante pour exécuter cette transformation en ligne de commandes :

```
tridentctl-protect create snapshotrestore <restore_name> \
  --snapshot <namespace/snapshot-name> \
  --namespace-mapping <source-ns>:<dest-ns> \
  --transformation
',v1,ConfigMap:remove{"path":"/metadata/annotations/kubectl.kubernetes.
io~1last-applied-configuration"}'
```

Remplacer une valeur dans une ressource

Utilisez l' `replace`opération pour remplacer une valeur existante dans une ressource au chemin spécifié. Le chemin JSON doit déjà exister.

Utiliser une CR

L'exemple suivant modifie le nom d'hôte d'un objet `Route` :

```
transformations:
  - resource:
      kind: Route
      group: route.openshift.io
    operations:
      - op: replace
        path: "/spec/host"
        value: "prod.example.com"
```

Utiliser l'interface de ligne de commande

Utilisez la commande suivante pour exécuter cette transformation en ligne de commandes :

```
tridentctl-protect create snapshotrestore <restore_name> \
  --snapshot <namespace/snapshot-name> \
  --namespace-mapping <source-ns>:<dest-ns> \
  --transformation
'route.openshift.io,v1,Route:replace{"path":"/spec/host","value":"prod.
example.com"}'
```

Tester la modification de la ressource

Utilisez l' `test`opération pour vérifier que la valeur à un chemin correspond à une valeur attendue avant que les autres opérations ne soient appliquées. Si le test échoue, la modification entière est annulée.

Utiliser une CR

L'exemple suivant met à jour `database-host` uniquement si `environment` est défini sur `staging` :

```
transformations:
  - resource:
      kind: ConfigMap
    operations:
      - op: test
        path: "/data/environment"
        value: "staging"
      - op: replace
        path: "/data/database-host"
        value: "prod-db.example.com"
```

Utiliser l'interface de ligne de commande

Utilisez la commande suivante pour exécuter cette transformation en ligne de commandes :

```
tridentctl-protect create snapshotrestore <restore_name> \
  --snapshot <namespace/snapshot-name> \
  --namespace-mapping <source-ns>:<dest-ns> \
  --transformation
',v1,ConfigMap:test{"path":"/data/environment","value":"staging"},repla
ce{"path":"/data/database-host","value":"prod-db.example.com"}'
```

Répliquez les applications à l'aide de NetApp SnapMirror et Trident Protect.

Avec Trident Protect, vous pouvez utiliser les capacités de réplication asynchrone de la technologie NetApp SnapMirror pour répliquer les données et les modifications d'applications d'un système de stockage à un autre, sur le même cluster ou entre différents clusters.

Annotations et étiquettes de namespace pendant les opérations de restauration et de basculement

Lors des opérations de restauration et de basculement, les libellés et les annotations dans l'espace de noms de destination correspondent aux libellés et aux annotations dans l'espace de noms source. Des étiquettes ou des annotations provenant de l'espace de noms source qui n'existent pas dans l'espace de noms de destination sont ajoutées et toutes les étiquettes ou annotations qui existent déjà sont écrasées pour correspondre à la valeur de l'espace de noms source. Les libellés ou annotations qui existent uniquement dans l'espace de noms de destination restent inchangés.

REMARQUE	Si vous utilisez Red Hat OpenShift, il est important de noter le rôle essentiel des annotations d'espace de noms dans les environnements OpenShift. Les annotations d'espace de noms garantissent que les pods restaurés adhèrent aux autorisations et aux configurations de sécurité appropriées définies par les contraintes de contexte de sécurité (SCC) OpenShift et peuvent accéder aux volumes sans problèmes d'autorisation. Pour plus d'informations, reportez-vous à la"Documentation sur les contraintes de contexte de sécurité OpenShift" .

Vous pouvez empêcher l'écrasement d'annotations spécifiques dans l'espace de noms de destination en configurant la variable d'environnement Kubernetes `RESTORE_SKIP_NAMESPACE_ANNOTATIONS` avant d'effectuer l'opération de restauration ou de basculement. Par exemple :

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect \
  --set-string
restoreSkipNamespaceAnnotations="{<annotation_key_to_skip_1>,<annotation_k
ey_to_skip_2>}" \
  --reuse-values
```

REMARQUE	Lors d'une opération de restauration ou de basculement, toutes les annotations et étiquettes d'espace de noms spécifiées dans `restoreSkipNamespaceAnnotations` et `restoreSkipNamespaceLabels` sont exclues de l'opération de restauration ou de basculement. Assurez-vous que ces paramètres sont configurés lors de l'installation initiale de Helm. Pour en savoir plus, consultez "Configurer les paramètres supplémentaires du graphique de barre de Trident Protect".

Si vous avez installé l'application source à l'aide de Helm avec le `--create-namespace` Le drapeau, un traitement spécial est accordé au `name` Légende. Lors du processus de restauration ou de basculement, Trident Protect copie cette étiquette dans l'espace de noms de destination, mais met à jour la valeur avec la valeur de l'espace de noms de destination si la valeur de la source correspond à l'espace de noms source. Si cette valeur ne correspond pas à l'espace de noms source, elle est copiée dans l'espace de noms de destination sans modification.

Exemple

L'exemple suivant présente un espace de noms source et de destination, chacun avec des annotations et des libellés différents. Vous pouvez voir l'état de l'espace de noms de destination avant et après l'opération, ainsi que la manière dont les annotations et les étiquettes sont combinées ou écrasées dans l'espace de noms de destination.

Avant l'opération de restauration ou de basculement

Le tableau suivant illustre l'état de l'exemple d'espaces de noms source et de destination avant l'opération de restauration ou de basculement :

Espace de noms	Annotations	Étiquettes
Espace de noms ns-1 (source)	• annotation.one/key : « updatedvalue » • annotation.deux/touche : « vrai »	• environnement=production • conformité = hipaa • name=ns-1
Espace de noms ns-2 (destination)	• annotation.un/touche : « vrai » • annotation.trois/touche : « false »	• role=base de données

Après l'opération de restauration

Le tableau suivant illustre l'état de l'exemple d'espace de noms de destination après une opération de restauration ou de basculement. Certaines clés ont été ajoutées, d'autres ont été écrasées et le `name` libellé a été mis à jour pour correspondre à l'espace de noms de destination :

Espace de noms	Annotations	Étiquettes
Espace de noms ns-2 (destination)	• annotation.one/key : « updatedvalue » • annotation.deux/touche : « vrai » • annotation.trois/touche : « false »	• name=ns-2 • conformité = hipaa • environnement=production • role=base de données

REMARQUE	Vous pouvez configurer Trident Protect pour geler et dégeler les systèmes de fichiers pendant les opérations de protection des données. "Apprenez-en davantage sur la configuration du gel du système de fichiers avec Trident Protect.".

Crochets d'exécution lors des opérations de basculement et d'inversion

Lorsque vous utilisez la relation AppMirror pour protéger votre application, il existe des comportements spécifiques liés aux hooks d'exécution dont vous devez être conscient lors des opérations de basculement et d'inversion.

- Lors du basculement, les hooks d'exécution sont automatiquement copiés du cluster source vers le cluster de destination. Il n'est pas nécessaire de les recréer manuellement. Après le basculement, les hooks d'exécution sont présents dans l'application et s'exécuteront lors de toute action pertinente.
- Lors d'une resynchronisation inverse ou d'une inversion, tous les hooks d'exécution existants sur l'application sont supprimés. Lorsque l'application source devient l'application de destination, ces hooks d'exécution ne sont plus valides et sont supprimés pour empêcher leur exécution.

Pour en savoir plus sur les hooks d'exécution, consultez"Gérer les hooks d'exécution de Trident Protect" .

Configuration d'une relation de réplication

La configuration d'une relation de réplication implique les éléments suivants :

- Choisir la fréquence à laquelle Trident Protect doit prendre un instantané de l'application (qui inclut les ressources Kubernetes de l'application ainsi que les instantanés 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. Sur le cluster source, créez un AppVault pour l'application 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> -n
trident-protect
```

2. Sur le cluster source, 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: my-app-name
  namespace: my-app-namespace
spec:
  includedNamespaces:
    - namespace: my-app-namespace
      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 <my-app-name> --namespaces <namespaces-to-be-included> -n <my-app-namespace>
```

3. Vous pouvez également, si vous le souhaitez, prendre un instantané de l'application source sur le cluster source. Cet instantané sert de base à l'application sur le cluster de destination. Si vous ignorez cette étape, vous devrez attendre la prochaine capture d'écran planifiée pour obtenir une version récente. Pour créer un instantané à la demande, reportez-vous à "Créer un snapshot à la demande".

4. Sur le cluster source, créez la ressource personnalisée (CR) de planification de réplication :

Outre le planning ci-dessous, il est recommandé de créer un planning de snapshots quotidien distinct avec une période de conservation de 7 jours afin de conserver un snapshot commun entre les clusters ONTAP appairés. Cela garantit la disponibilité des snapshots pendant 7 jours maximum, mais la période de conservation peut être personnalisée selon les besoins de l'utilisateur.

REMARQUE

En cas de basculement, le système peut utiliser ces snapshots pendant sept jours maximum pour les opérations d'inversion. Cette approche rend le processus d'inversion plus rapide et plus efficace, car seules les modifications apportées depuis le dernier snapshot sont transférées, et non l'intégralité des données.

Si un calendrier existant pour l'application répond déjà aux exigences de conservation souhaitées, aucun calendrier supplémentaire n'est requis.

Créez la planification de réplication à l'aide d'une ressource personnalisée (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** : (*Obligatoire*) Cette valeur doit correspondre au champ metadata.name de l'AppVault pour l'application source.
   - **spec.applicationRef**: (*Obligatoire*) Cette valeur doit correspondre au champ metadata.name de la ressource personnalisée (CR) de l'application source.
   - **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
  namespace: my-app-namespace
spec:
  appVaultRef: my-appvault-name
  applicationRef: my-app-name
  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
```

Créez la planification de réplication à l'aide de l'interface de ligne de commande (CLI).

a. Créez la planification de réplication en remplaçant les valeurs entre crochets par les informations provenant de votre environnement :

```
tridentctl-protect create schedule --name appmirror-schedule
--app <my_app_name> --appvault <my_app_vault> --granularity
Custom --recurrence-rule <rule> --snapshot-retention
<snapshot_retention_count> -n <my_app_namespace>
```

Exemple:

```
tridentctl-protect create schedule --name appmirror-schedule
--app <my_app_name> --appvault <my_app_vault> --granularity
Custom --recurrence-rule  "DTSTART:20220101T000200Z
\nRRULE:FREQ=MINUTELY;INTERVAL=5" --snapshot-retention 2 -n
<my_app_namespace>
```

5. Sur le cluster de destination, créez une application source AppVault CR identique à la CR AppVault que vous avez appliquée sur le cluster source et nommez-la (par exemple, `trident-protect-appvault-primary-destination.yaml`).
6. Appliquer la CR :

```
kubectl apply -f trident-protect-appvault-primary-destination.yaml -n
trident-protect
```

7. Créez une CR AppVault de destination 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 trident-protect
```

8. Sur le cluster de destination, créez un fichier CR AppMirrorRelationship.

REMARQUE	Lors de l'utilisation d'une modification commerciale (CR), créez manuellement l'espace de noms de destination avant d'appliquer la CR. Trident Protect crée automatiquement des espaces de noms uniquement lors de l'utilisation de l'interface de ligne de commande (CLI).

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.sourceApplicationUID** : (Obligatoire) Cette valeur doit correspondre à l'UID de l'application source que vous avez définie dans la CR de l'application source.
- **spec.storageClassName**: (*Optionnel*) Choisissez le nom d'une classe de stockage valide sur le cluster. La classe de stockage doit être liée à une machine virtuelle de stockage ONTAP qui est appariée avec l'environnement source. Si la classe de stockage n'est pas spécifiée, la classe de stockage par défaut du cluster sera utilisée.
- **Spécifications storageClassMapping** : (Optionnel) Associe les classes de stockage source aux classes de stockage de destination pour chaque PVC au sein du même AMR. Chaque entrée nécessite un `source` et un `destination` nom de classe de stockage. Lorsqu'elle est définie, `storageClassMapping` remplace `storageClassName` entièrement ; les PVC sans entrée source correspondante utilisent la classe de stockage par défaut du cluster, et non `storageClassName`.

  Lors de l'utilisation `storageClassMapping` dans un CR AppMirrorRelationship, veuillez prendre en compte les restrictions suivantes :

  - `storageClassMapping`remplace `storageClassName`entièrement. Lorsque les deux champs sont renseignés, `storageClassName`est ignoré.
  - Les PVC avec des classes de stockage source qui n'ont pas d'entrée correspondante dans `storageClassMapping` utilisent la classe de stockage par défaut du cluster, pas `storageClassName`.
  - Chaque entrée nécessite à la fois une valeur `source` et `destination`.
  - Modifier la classe de stockage de destination peut entraîner la suppression et la recréation des PVC de destination existants.
- **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: my-app-name
  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 crochets par les informations provenant de votre environnement :

```
tridentctl-protect create appmirrorrelationship
<name_of_appmirorrelationship> --destination-app-vault
<my_vault_name> --source-app-vault <my_vault_name> --recurrence
-rule <rule> --namespace-mapping <ns_mapping> --source-app-id
<source_app_UID> --source-app <my_source_app_name> --storage
-class <storage_class_name> --storageclass-mapping
<src_class:dst_class> -n <application_namespace>
```

REMARQUE	Utilisez `--storageclass-mapping` pour associer des classes de stockage source individuelles à des classes de stockage de destination. Spécifiez une ou plusieurs `source:destination` paires séparées par des virgules (par exemple, `"class-a:class-b,class-c:class-d"`). Utilisez `*:destination-class` comme caractère générique pour associer tous les PVC non mappés à une seule classe de stockage de destination. Lorsque `--storageclass-mapping` est défini, `--storage-class` est ignoré.

Exemple utilisant une seule classe de stockage de destination (`--storage-class`):

```
tridentctl-protect create appmirrorrelationship my-amr
--destination-app-vault appvault2 --source-app-vault appvault1
--recurrence-rule
"DTSTART:20220101T000200Z\nRRULE:FREQ=MINUTELY;INTERVAL=5"
--source-app my-app --namespace-mapping "my-source-ns1:my-dest-
ns1,my-source-ns2:my-dest-ns2" --source-app-id 373f24c1-5769-
404c-93c3-5538af6ccc36 --storage-class my-storage-class -n my-
dest-ns1
```

9. (*Optional*) sur le cluster de destination, 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

Avec Trident Protect, vous pouvez basculer les applications répliquées vers un cluster de destination. Cette procédure interrompt 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 elle était opérationnelle.

Étapes

1. Sur le cluster de destination, modifiez 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.

IMPORTANT	Toutes les données écrites sur l'application de destination pendant le basculement sont perdues.

Étapes

1. Facultatif : sur le cluster source, créez un snapshot de l'application source. Cela permet de s'assurer que les dernières modifications du cluster source sont capturées.
2. Sur le cluster de destination, modifiez 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. Sur le cluster de destination d'origine, supprimez la CR AppMirrorRelationship. 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 la nouvelle destination (cluster source d'origine) est configurée avec les deux CRS AppVault.
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 système de

stockage de destination tout en continuant à répliquer vers le système de stockage source d'origine. Trident Protect arrête l'application source et réplique les données vers la destination avant de basculer vers l'application de destination.

Dans ce cas, vous permutez la source et la destination.

Étapes

1. Sur le cluster source, créer un snapshot 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: my-app-name
```

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> -n
<application_namespace>
```

2. Sur le cluster source, une fois l'instantané d'arrêt terminé, obtenir l'état de l'instantané d'arrêt :

```
kubectl get shutdownsnapshot -n my-app-namespace
<shutdown_snapshot_name> -o yaml
```

3. Sur le cluster source, 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 nouveau cluster de destination vers le nouveau cluster source, avec la modification suivante :

REMARQUE	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

Avec Trident Protect, vous pouvez effectuer un « retour en arrière » après une opération de basculement en utilisant la séquence d'opérations suivante. Dans ce flux de travail visant à rétablir le sens de réplication d'origine, Trident Protect réplique (resynchronise) toutes les modifications apportées à l'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.

AVERTISSEMENT	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. Sur le cluster de désaturation actuel, supprimez la CR AppMirrorRelationship :

```
kubectl delete -f trident-protect-relationship.yaml -n my-app-namespace
```

Migrer les applications à l'aide de Trident Protect

Vous pouvez migrer vos applications entre clusters ou vers différentes classes de stockage en restaurant les données de sauvegarde.

REMARQUE	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 cluster différent (effectuer un clonage inter-clusters), créez une sauvegarde sur le cluster source, puis restaurez la sauvegarde sur un cluster différent. Assurez-vous que Trident Protect est installé sur le cluster de destination.

REMARQUE	Vous pouvez répliquer une application entre différents clusters à l'aide de "Réplication SnapMirror".

Étapes

1. "Créer une sauvegarde".
2. Assurez-vous que la CR AppVault du compartiment de stockage objet contenant la sauvegarde a été configurée sur le cluster de destination.
3. Sur le cluster de destination, "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 une sauvegarde vers la classe de stockage de destination.

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

Étapes

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 :
     - **ResourceFilter.resourceMatchers** : un tableau d'objets resourceMatcher. Si vous définissez plusieurs éléments dans ce tableau, ils correspondent en tant qu'opération OU et les champs de chaque élément (groupe, type, version) correspondent en tant qu'opération ET.
       - **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-1
        kind: my-resource-kind-1
        version: my-resource-version-1
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
      - group: my-resource-group-2
        kind: my-resource-kind-2
        version: my-resource-version-2
        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

Étapes

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>
```

Protégez les machines virtuelles KubeVirt avec Trident Protect

Trident Protect vous permet de sauvegarder et de restaurer des machines virtuelles (VM) KubeVirt exécutées sur OpenShift Virtualization. Vous pouvez protéger toutes les VM d'un ou de plusieurs espaces de noms à la fois, ou utiliser le champ `includedVirtualMachines` pour cibler des VM spécifiques par leur nom. Trident Protect collecte automatiquement toutes les ressources VM dépendantes lors de la sauvegarde, et lors de la restauration, vous pouvez récupérer l'intégralité de l'archive ou restaurer sélectivement des VM individuelles.

KubeVirt modes de protection

Trident Protect prend en charge deux méthodes pour définir quelles KubeVirt VMs protéger. Choisissez un mode lors de la définition d'une application :

Mode	Champ	Description
Basé sur les espaces de noms	`includedNamespaces`	Protégez toutes les ressources correspondantes dans un ou plusieurs espaces de noms. Utilisez un `labelSelector` sur l'entrée d'espace de noms pour contrôler quelles machines virtuelles sont incluses dynamiquement.
à l'échelle de la VM	`includedVirtualMachines`	Protégez des machines virtuelles KubeVirt spécifiques par espace de noms et par nom. Utilisez ce mode lorsque les machines virtuelles sont réparties sur plusieurs espaces de noms ou lorsque vous ne souhaitez pas protéger toutes les ressources d'un espace de noms.

Vous devez spécifier exactement un seul champ (`includedNamespaces` ou `includedVirtualMachines`) dans le `ApplicationSpec`. Vous ne pouvez pas utiliser les deux dans la même application.

Les deux modes prennent en charge un filtrage supplémentaire :

- Utilisez `resourceFilter` pour inclure ou exclure des ressources spécifiques.
- Utilisez `includedClusterScopedResources` pour inclure des ressources compatibles à l'échelle du cluster, telles que des objets `StorageClass`.

REMARQUE Trident Protect peut geler et dégeler les systèmes de fichiers des machines virtuelles KubeVirt lors des opérations de protection des données afin de garantir leur cohérence. Pour plus de détails sur la configuration, consultez "Protéger les données avec les machines virtuelles KubeVirt".

Ressources collectées lors de la sauvegarde et de l'instantané

Lorsque vous créez une sauvegarde ou un instantané pour une application qui inclut des machines virtuelles KubeVirt, Trident Protect collecte les ressources de machines virtuelles sélectionnées et découvre automatiquement leurs ressources dépendantes.

Le comportement de collecte s'applique dans deux cas :

- Applications à portée de machine virtuelle qui utilisent `includedVirtualMachines`.
- Applications basées sur des espaces de noms qui contiennent KubeVirt `VirtualMachine` resources.

collection à portée de VM

Pour les applications à portée de machine virtuelle, Trident Protect commence par chaque `VirtualMachine` répertoriée dans `includedVirtualMachines` et découvre les ressources dépendantes nécessaires à la restauration de cette machine virtuelle.

Type de ressource	Portée	Comment il est découvert
`VirtualMachine`	Associé à un espace de noms	Correspondance directe à partir de la `includedVirtualMachines` liste.
`VirtualMachineInstance`	Associé à un espace de noms	Même espace de noms et même nom que la machine virtuelle, si la machine virtuelle est en cours d'exécution.
`VirtualMachineInstancetype`	Associé à un espace de noms	Référencé par `spec.instancetype` lorsque `kind: VirtualMachineInstancetype`.
`VirtualMachineClusterInstancetype`	Cluster	Référencé par `spec.instancetype` lorsque `kind` n'est pas défini ou `kind: VirtualMachineClusterInstancetype`.
`VirtualMachinePreference`	Associé à un espace de noms	Référencé par `spec.preference` lorsque `kind: VirtualMachinePreference`.
`VirtualMachineClusterPreference`	Cluster	Référencé par `spec.preference` lorsque `kind` n'est pas défini ou `kind: VirtualMachineClusterPreference`.
`DataVolume`	Associé à un espace de noms	Cité par `spec.template.spec.volumes[].dataVolume.name`.
`PersistentVolumeClaim`	Associé à un espace de noms	Découvert à partir des PVC de support DataVolume, qui utilisent le même nom que le DataVolume ; références PVC directes dans `persistentVolumeClaim.claimName` ; PVC de vidage mémoire dans `memoryDump.claimName` ; et PVC éphémères dans `ephemeral.persistentVolumeClaim.claimName`.
`PersistentVolume`	Cluster	Découvert à partir du support PV pour chaque PVC collecté en utilisant `spec.volumeName`.

Type de ressource	Portée	Comment il est découvert
`Secret`	Associé à un espace de noms	Découverts à partir des secrets de volume dans `secret.secretName` ; secrets d'extraction d'images containerDisk ; références secrètes de données utilisateur et de données réseau CloudInitNoCloud et CloudInitConfigDrive ; secrets Sysprep ; secrets d'extraction d'images de conteneurs KernelBoot ; et secrets SSH ou de mot de passe AccessCredentials.
`ConfigMap`	Associé à un espace de noms	Découvert à partir du volume ConfigMaps dans `configMap.name` et Sysprep ConfigMaps.
`ServiceAccount`	Associé à un espace de noms	Découvert à partir de `serviceAccount.serviceAccountName`références de volumes.
`NetworkAttachmentDefinition`	Associé à un espace de noms	Découvert à partir des réseaux Multus `spec.template.spec.networks[].multus.networkName` lorsque le nom du réseau ne contient pas `/`. Seules les références NetworkAttachmentDefinition du même espace de noms sont collectées. Les références entre espaces de noms différents sont ignorées.
`Pod`	Associé à un espace de noms	Découvert pour `virt-launcher` les Pods en utilisant le sélecteur d'étiquettes `kubevirt.io=virt-launcher,vm.kubevirt.io/name=<vmName>`.
Classe de stockage, volumes Trident et backends Trident	Cluster	Collectées de la même manière que les applications basées sur les espaces de noms, en fonction des PVC découverts pour la VM.

REMARQUE Une fois toutes les ressources d'espace de noms au niveau de la machine virtuelle collectées, Trident Protect applique la méthode standard `resourceFilter`, si elle est configurée. La collecte des ressources au niveau du cluster se déroule ensuite normalement.

Collection basée sur les espaces de noms

Pour les applications basées sur des espaces de noms, Trident Protect analyse les espaces de noms sélectionnés et collecte toutes les ressources correspondantes. Si des KubeVirt `VirtualMachine` objets sont trouvés, Trident Protect collecte automatiquement leurs ressources dépendantes, notamment les types d'instances, les préférences et `virt-launcher` Pods, de la même manière que pour les applications à portée de machine virtuelle.

Cela signifie que les ressources spécifiques à la VM sont incluses dans votre sauvegarde ou votre instantané même lorsque l'application est définie avec `includedNamespaces`.

Protégez les machines virtuelles à l'aide de définitions d'applications basées sur les espaces de noms

Lors de la création d'une application basée sur un espace de noms, Trident Protect analyse les espaces de noms sélectionnés et collecte toutes les ressources, y compris toutes les machines virtuelles KubeVirt. Pour chaque machine virtuelle détectée, Trident Protect collecte également les ressources dépendantes nécessaires à la sauvegarde et à la restauration de cette machine virtuelle.

Exemple de CR d'application basé sur un espace de noms :

```
apiVersion: protect.trident.netapp.io/v1
kind: Application
metadata:
  name: <application_name>
  namespace: <application_namespace>
spec:
  includedNamespaces:
    - namespace: <vm_namespace>
```

Après avoir créé l'application, utilisez les workflows standard de Trident Protect pour les sauvegardes et les instantanés :

- Pour des exemples de YAML d'instantané, voir "Créer un snapshot à la demande".
- Pour des exemples de sauvegarde YAML, voir "Créez une sauvegarde à la demande".

Comportement de restauration pour les applications de machines virtuelles basées sur les espaces de noms

Étant donné que les applications basées sur les espaces de noms collectent les ressources des espaces de noms sélectionnés, la sauvegarde ou l'instantané peut inclure des objets d'exécution KubeVirt et des métadonnées gérées par KubeVirt. Certains de ces objets ne doivent pas être restaurés directement, car ils représentent l'état d'exécution du cluster source.

Lors de la restauration, Trident Protect applique des transformations compatibles KubeVirt pour supprimer les métadonnées du cluster source et ignorer les ressources transitoires KubeVirt. Cela aide KubeVirt à recréer la machine virtuelle proprement dans l'environnement restauré.

Ressource	Transformer
VirtualMachine	Supprime les étiquettes et annotations qui commencent par `kubevirt.io/`.
VirtualMachineInstance	Supprime les étiquettes et annotations qui commencent par `kubevirt.io/`.
VirtualMachineInstanceMigration	Ignoré et non restauré.
VirtualMachineSnapshot	Ignoré et non restauré.
VirtualMachineSnapshotContent	Ignoré et non restauré.
Pod (`virt-launcher`)	Ignoré lorsque le pod possède le label `kubevirt.io=virt-launcher`.

REMARQUE Trident Protect restaure les ressources KubeVirt dans un ordre défini afin d'éviter les problèmes de dépendance. Les CDI DataVolumes (`cdi.kubevirt.io` sont restaurés en premier afin que les données disque de la VM soient disponibles avant la création des objets VM. Ensuite, les VirtualMachines (`kubevirt.io` sont restaurées avant les VirtualMachineInstances.

Protection dynamique des machines virtuelles avec sélecteurs d'étiquettes

Si vous souhaitez protéger les machines virtuelles de manière dynamique sans avoir à lister le nom de chaque

machine virtuelle, utilisez des définitions d'application basées sur l'espace de noms avec des sélecteurs d'étiquettes.

Seule la ressource VirtualMachine CR nécessite des étiquettes. Les ressources dépendantes telles que DataVolumes, les PVC, les Secrets et les ConfigMaps sont collectées automatiquement.

Étapes

1. Étiquetez les CR de machines virtuelles avec une étiquette partagée :

```
kubectl label virtualmachine <vm_name 1> app=<label_value> -n
<vm_namespace>
kubectl label virtualmachine <vm_name_2> app=<label_value> -n
<vm_namespace>
```

2. Créez un CR d'application avec `includedNamespaces` et un `labelSelector` :

```
apiVersion: protect.trident.netapp.io/v1
kind: Application
metadata:
  name: <application_name>
  namespace: <application_namespace>
spec:
  includedNamespaces:
    - namespace: <vm_namespace>
      labelSelector:
        matchLabels:
          app: <label_value>
```

3. Créez des sauvegardes et des restaurations à l'aide du workflow standard Trident Protect.

Détails supplémentaires :

- Vous pouvez ajouter ou supprimer des machines virtuelles de la protection à tout moment en modifiant les étiquettes :

```
kubectl label virtualmachine <vm_name_to_add> app=<label_value> -n
<vm_namespace>
kubectl label virtualmachine <vm_name_to_remove> app
=<different_label_value> -n <vm_namespace>
```

- Vous pouvez également utiliser des sélecteurs d'étiquettes dans plusieurs espaces de noms :

```
spec:
  includedNamespaces:
    - namespace: <vm_namespace_1>
      labelSelector:
        matchLabels:
          <label_key>: <label_value>
    - namespace: <vm_namespace_2>
      labelSelector:
        matchLabels:
          <label_key>: <label_value>
```

Protégez des machines virtuelles spécifiques avec includedVirtualMachines

Utilisez `includedVirtualMachines` lorsque vous souhaitez protéger un ensemble nommé de VirtualMachines sur un ou plusieurs espaces de noms et restaurer sélectivement des machines virtuelles spécifiques à partir d'un archivage des sauvegardes.

Les demandes de modification (CR) d'application et de restauration utilisent le même `includedVirtualMachines` format de sélecteur.

Utilisez ce sélecteur dans différents contextes :

- Dans une demande de modification d'application, elle définit quelles machines virtuelles sont protégées.
- Dans une demande de restauration (CR), elle définit quelles machines virtuelles sont restaurées à partir de l'archivage des sauvegardes ou de l'archive de snapshot.

REMARQUE	Lors de la restauration des CR, `includedVirtualMachines` est facultatif. Si vous ne renseignez pas ce champ, l'archive entière est restaurée. Lorsqu'il est spécifié, seules les machines virtuelles listées et leurs ressources dépendantes sont restaurées.

Le code YAML ci-dessous illustre la structure du sélecteur partagé :

```
includedVirtualMachines:
  - namespace: <vm_namespace_1>
    names:
      - <vm_name_1>
      - <vm_name_2>
  - namespace: <vm_namespace_2>
    names:
      - <vm_name_3>
```

REMARQUE	Plusieurs entrées dans `includedVirtualMachines` peuvent faire référence à différents espaces de noms, ce qui permet des définitions et des restaurations d'applications compatibles avec les machines virtuelles multi-espaces de noms.

Champ	Type	Description
espace de noms	chaîne	Espace de noms Kubernetes contenant les objets de la machine virtuelle cible.
noms	liste de chaînes de caractères	Un ou plusieurs noms de machines virtuelles dans cet espace de noms.

Exemple d'application basée sur une machine virtuelle :

```
apiVersion: protect.trident.netapp.io/v1
kind: Application
metadata:
  name: <application_name>
  namespace: <application_namespace>
spec:
  includedVirtualMachines:
    - namespace: <vm_namespace_1>
      names:
        - <vm_name_1>
        - <vm_name_2>
    - namespace: <vm_namespace_2>
      names:
        - <vm_name_3>
  includedClusterScopedResources:
    - group: storage.k8s.io
      kind: StorageClass
```

Restaurer des machines virtuelles KubeVirt spécifiques

Vous pouvez utiliser `includedVirtualMachines` sur les CR de restauration pour restaurer uniquement les machines virtuelles KubeVirt sélectionnées à partir d'une sauvegarde, d'un instantané ou d'une archive d'instantanés répliqués.

Lorsque `includedVirtualMachines` est spécifié sur un CR de restauration, Trident Protect ne restaure que les VirtualMachines listées et leurs ressources dépendantes, telles que `VirtualMachineInstance`, `DataVolume`, `PersistentVolumeClaim`, `PersistentVolume`, `Secret`, `ConfigMap`, `ServiceAccount` et d'autres objets liés aux VM.

Si vous ne spécifiez `includedVirtualMachines` pas dans la CR de restauration, Trident Protect restaure l'archive complète conformément à la configuration de la CR de restauration.

CR et commandes CLI de restauration prises en charge

Le tableau suivant répertorie les CR de restauration qui prennent en charge `includedVirtualMachines` et leurs commandes CLI correspondantes.

Restaurer CR	Commande CLI	Description
`BackupRestore`	`tridentctl-protect create backuprestore`	Restaure à partir d'une sauvegarde vers des mappages d'espaces de noms nouveaux ou différents. Prend en charge `namespaceMapping` et `storageClassMapping`.
`BackupInplaceRestore`	`tridentctl-protect create backupinplacerestore`	Restaure à partir d'une sauvegarde dans l'espace de noms ou les espaces de noms d'origine.
`SnapshotRestore`	`tridentctl-protect create snapshotrestore`	Restaure à partir d'un instantané vers des mappages d'espaces de noms nouveaux ou différents. Prend en charge `namespaceMapping` et `storageClassMapping`.
`SnapshotInplaceRestore`	`tridentctl-protect create snapshotinplacerestore`	Restaure à partir d'un instantané dans l'espace de noms ou les espaces de noms d'origine.
`ReplicateSnapshotRestore`	YAML uniquement, ou utilisez la commande CLI prise en charge si elle est disponible dans votre environnement	Restaure à partir d'un instantané répliqué. Prend en charge `namespaceMapping`, `storageClassMapping`, ainsi que la restauration sur place à l'aide du drapeau `inPlaceRestore`.

REMARQUE L' `tridentctl-protect` CLI ne propose pas de paramètre pour `includedVirtualMachines` les commandes de restauration. Pour restaurer des machines virtuelles spécifiques, vous devez appliquer directement un fichier YAML de restauration CR à l'aide de `kubectl apply`. Vous pouvez rédiger le fichier YAML depuis zéro, ou utiliser la CLI avec `--dry-run --output yaml` pour générer un fichier de départ, ajouter le champ `includedVirtualMachines` sous `spec`, puis l'appliquer.

Comportement de restauration à l'échelle de la machine virtuelle

Le `includedVirtualMachines`champ d'une CR de restauration utilise le même format de sélecteur que la CR d'application :

```
includedVirtualMachines:
  - namespace: <source_vm_namespace>
    names:
      - <vm_name_1>
      - <vm_name_2>
```

Pour les CR de restauration, les valeurs d'espace de noms dans `includedVirtualMachines` font référence aux espaces de noms sources tels qu'elles existent dans l'archive de sauvegarde, l'instantané ou l'archive d'instantané répliqué.

Si vous utilisez `namespaceMapping`, Trident Protect sélectionne d'abord les machines virtuelles demandées dans l'archive source, puis applique les mappages d'espace de noms lors de la restauration. Si vous utilisez

`storageClassMapping`, Trident Protect applique les mappages de classe de stockage lors de la phase de transformation de la restauration.

Générez le fichier YAML de restauration avec l'interface de ligne de commande (CLI)

Vous pouvez utiliser `tridentctl-protect` pour générer un CR de restauration de base, puis modifier le YAML pour ajouter `includedVirtualMachines`.

```
tridentctl-protect create <restore_command> <restore_name> \
  <restore_options> \
  --dry-run --output yaml > <restore_name>.yaml
```

Modifiez le fichier YAML généré et ajoutez le champ `includedVirtualMachines` sous `spec`, puis appliquez le fichier :

```
kubectl apply -f <restore_name>.yaml
```

Exemple : Définir une application basée sur une machine virtuelle

L'application suivante protège trois machines virtuelles spécifiques réparties sur deux espaces de noms.

```
apiVersion: protect.trident.netapp.io/v1
kind: Application
metadata:
  name: production-vms
  namespace: prod
spec:
  includedVirtualMachines:
    - namespace: prod
      names:
        - app-server
        - database-server
    - namespace: monitoring
      names:
        - prometheus-vm
  includedClusterScopedResources:
    - group: storage.k8s.io
      kind: StorageClass
```

Exemple : créer une sauvegarde

La création d'une sauvegarde pour une application basée sur une machine virtuelle utilise le CR de sauvegarde standard. Aucun champ spécifique à la machine virtuelle n'est requis dans le CR de sauvegarde.

```yaml
apiVersion: protect.trident.netapp.io/v1
kind: Backup
metadata:
  name: daily-backup
  namespace: prod
spec:
  applicationRef: production-vms
  appVaultRef: s3-vault
```

Exemple : Restaurer une seule machine virtuelle à partir d'une sauvegarde

L'exemple suivant restaure uniquement `database-server` à partir de l'archivage des sauvegardes. Les autres machines virtuelles de l'archive, telles que `app-server` et `prometheus-vm`, sont ignorées.

```yaml
apiVersion: protect.trident.netapp.io/v1
kind: BackupRestore
metadata:
  name: restore-db-only
  namespace: prod-vms-restore
spec:
  appVaultRef: s3-vault
  appArchivePath: backups/production-vms/2026-03-10T00-00-00Z
  namespaceMapping:
    - source: prod
      destination: prod-dr
  includedVirtualMachines:
    - namespace: prod
      names:
        - database-server
```

```
kubectl apply -f restore-db-only.yaml
```

Seuls `database-server` et ses ressources dépendantes, telles que DataVolumes, les PVC, les PV, les Secrets, les ConfigMaps et les ressources associées KubeVirt, sont restaurés.

Exemple : Restaurer une seule machine virtuelle sur place à partir d'une sauvegarde

L'exemple suivant restaure uniquement `database-server` dans son espace de noms d'origine à partir de l'archivage des sauvegardes.

```
apiVersion: protect.trident.netapp.io/v1
kind: BackupInplaceRestore
metadata:
  name: ipr-db-only
  namespace: prod
spec:
  appVaultRef: s3-vault
  appArchivePath: backups/production-vms/2026-03-10T00-00-00Z
  includedVirtualMachines:
    - namespace: prod
      names:
        - database-server
```

```
kubectl apply -f ipr-db-only.yaml
```

Exemple : Restaurer une seule machine virtuelle à partir d'un instantané

L'exemple suivant restaure uniquement `ubuntu-vm-blue-a627be38` à partir d'une archive de snapshot.

```
apiVersion: protect.trident.netapp.io/v1
kind: SnapshotRestore
metadata:
  name: blue-vm-snap-restore
  namespace: blue
spec:
  appVaultRef: my-appvault
  appArchivePath: snapshots/protectctl-blue-vm/snap-1
  includedVirtualMachines:
    - namespace: blue
      names:
        - ubuntu-vm-blue-a627be38
```

```
kubectl apply -f blue-vm-snap-restore.yaml
```

Combinez la restauration de la machine virtuelle avec d'autres filtres

Vous pouvez utiliser `includedVirtualMachines` avec `resourceFilter`, `namespaceMapping` et `storageClassMapping` dans le même CR de restauration pour affiner ce qui est restauré.

Lorsque vous ajoutez un `resourceFilter`, Trident Protect sélectionne d'abord les machines virtuelles listées dans `includedVirtualMachines` et leurs ressources dépendantes. Il applique ensuite le filtre de ressources pour affiner ce qui est inclus ou exclu de ces résultats.

```
spec:
  includedVirtualMachines:
    - namespace: <vm_namespace>
      names:
        - <vm_name>
  resourceFilter:
    resourceSelectionCriteria: Include
    resourceMatchers:
      - kinds: ["PersistentVolumeClaim"]
        names: ["<pvc_name>"]
```

Pour `BackupRestore`, `SnapshotRestore` et `ReplicateSnapshotRestore`, `namespaceMapping` et `storageClassMapping` fonctionnent conjointement avec `includedVirtualMachines`. Les références d'espace de noms dans `includedVirtualMachines` correspondent aux espaces de noms sources tels qu'elles apparaissent dans l'archive. Les mappages d'espace de noms et de classe de stockage sont appliqués lors de la phase de transformation de restauration, une fois le filtrage terminé.

Pour plus de détails sur le comportement de restauration, veuillez vous référer à "Restaurez les applications à l'aide de Trident Protect".

Créez une application à portée de machine virtuelle à l'aide de l'interface de ligne de commande (CLI)

Utilisez le `--virtual-machines` indicateur pour limiter la portée d'une application à des machines virtuelles KubeVirt spécifiques plutôt qu'à des espaces de noms entiers. Le `--virtual-machines` indicateur est mutuellement exclusif avec `--namespaces`.

Étapes

1. Créez l'application en exécutant l'une des commandes suivantes, en remplaçant les valeurs entre crochets par les informations de votre environnement. Le format pour l' `--virtual-machines`option est `<namespace>(<vm_name>)`. Pour plusieurs machines virtuelles, utilisez des noms séparés par des virgules entre parenthèses :
   - Une seule machine virtuelle dans un espace de noms :

     ```
     tridentctl-protect create application <application_name> \
       -n <application_namespace> \
       --virtual-machines "<vm_namespace>(<vm_name>)"
     ```

   - Plusieurs machines virtuelles réparties dans différents espaces de noms :

     ```
     tridentctl-protect create application <application_name> \
       -n <application_namespace> \
       --virtual-machines "<vm_namespace_1>(<vm_name_1>,<vm_name_2>
     ),<vm_namespace_2>(<vm_name_3>)"
     ```

Informations connexes

- "Définissez une application pour la gestion"
- "Protégez les applications"
- "Restaurez les applications à l'aide de Trident Protect"
- "Utilisez les paramètres de restauration avancés"
- "Installez le plugin CLI Trident Protect"
- "Utilisez les objets AppVault pour gérer les compartiments"

Gérer les hooks d'exécution de Trident Protect

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 hooks 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. Des blocages du système de fichiers se produisent, le cas échéant. "Apprenez-en davantage sur la configuration du gel du système de fichiers avec Trident Protect.".
3. L'opération de protection des données est effectuée.
4. Les systèmes de fichiers gelés ne sont pas gelés, le cas échéant.
5. 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

REMARQUE	L'exemple d'ordre précédent s'applique uniquement lorsque vous exécutez une sauvegarde qui n'utilise pas de snapshot existant.
REMARQUE	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.
REMARQUE	Si un hook d'exécution pré-snapshot ajoute, modifie ou supprime des ressources Kubernetes, ces modifications sont incluses dans l'instantané ou la sauvegarde et dans toute opération de restauration ultérieure.

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 hooks d'exécution soient écrits au format de scripts shell exécutables.
- La taille du script est limitée à 96 Ko.
- Trident Protect utilise les paramètres d'exécution et tous les critères correspondants pour déterminer quels hooks sont applicables à une opération de snapshot, de sauvegarde ou de restauration.

REMARQUE	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 hook d'exécution utilise une expression régulière pour faire correspondre les conteneurs de votre cluster. Lorsqu'un hook correspond à un conteneur, le hook exécutera son script associé sur ce conteneur. Les expressions régulières pour les filtres utilisent la syntaxe d'expression régulière 2 (RE2), qui ne prend pas en charge la création d'un filtre excluant 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, consultez "Prise en charge de la syntaxe de l'expression régulière 2 (RE2)" .

REMARQUE	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.

REMARQUE	Les scripts du dépôt Verda sont fournis tels quels et ne sont pas officiellement pris en charge par NetApp. Le support NetApp couvre uniquement le framework des hooks d'exécution au sein de Trident Protect.

Créer un crochet d'exécution

Vous pouvez créer un point d'exécution personnalisé pour une application utilisant Trident Protect. Vous devez disposer des autorisations de propriétaire, d'administrateur ou de membre pour créer des points d'exécution.

Utiliser une CR

Étapes

1. Créez le fichier de ressource personnalisée (CR) et nommez-le `trident-protect-hook.yaml`.
2. Configurez les attributs suivants pour qu'ils correspondent à votre environnement Trident Protect et à la configuration de votre 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

Étapes

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> -n <application_namespace>
```

Exécutez manuellement un crochet d'exécution

Vous pouvez exécuter manuellement un crochet d'exécution à des fins de test ou si vous devez exécuter de nouveau le crochet manuellement après un échec. Vous devez disposer des autorisations propriétaire, administrateur ou membre pour exécuter manuellement les crochets d'exécution.

L'exécution manuelle d'un crochet d'exécution se compose de deux étapes de base :

1. Créez une sauvegarde de ressource, qui collecte les ressources et crée une sauvegarde de celles-ci, en déterminant l'emplacement d'exécution du hook
2. Exécutez le crochet d'exécution contre la sauvegarde

Étape 1 : création d’une sauvegarde de ressource

Utiliser une CR

Étapes

1. Créez le fichier de ressource personnalisée (CR) et nommez-le `trident-protect-resource-backup.yaml`.
2. Configurez les attributs suivants pour qu'ils correspondent à votre environnement Trident Protect et à la configuration de votre 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 créer la sauvegarde de ressource.
   - **Spec.appVaultRef**: (*required*) Nom de l'AppVault où sont stockés le contenu de la sauvegarde.
   - **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}'
     ```

     Exemple YAML :

```
---
apiVersion: protect.trident.netapp.io/v1
kind: ResourceBackup
metadata:
  name: example-resource-backup
spec:
  applicationRef: my-app-name
  appVaultRef: my-appvault-name
  appArchivePath: example-resource-backup
```

3. Une fois que vous avez rempli le fichier CR avec les valeurs correctes, appliquez la CR :

```
kubectl apply -f trident-protect-resource-backup.yaml
```

Utiliser l'interface de ligne de commande

Étapes

1. Créez la sauvegarde en remplaçant les valeurs entre parenthèses par les informations de votre environnement. Par exemple :

```
tridentctl protect create resourcebackup <my_backup_name> --app
<my_app_name> --appvault <my_appvault_name> -n
<my_app_namespace> --app-archive-path <app_archive_path>
```

2. Afficher l'état de la sauvegarde. Vous pouvez utiliser cet exemple de commande plusieurs fois jusqu'à ce que l'opération soit terminée :

```
tridentctl protect get resourcebackup -n <my_app_namespace>
<my_backup_name>
```

3. Vérifiez que la sauvegarde a réussi :

```
kubectl describe resourcebackup <my_backup_name>
```

Étape 2 : exécutez le crochet d’exécution

Utiliser une CR

Étapes

1. Créez le fichier de ressource personnalisée (CR) et nommez-le `trident-protect-hook-run.yaml`.
2. Configurez les attributs suivants pour qu'ils correspondent à votre environnement Trident Protect et à la configuration de votre cluster :

   - **metadata.name**: (*required*) le nom de cette ressource personnalisée; choisissez un nom unique et sensible pour votre environnement.
   - **Spec.applicationRef**: (*required*) Assurez-vous que cette valeur correspond au nom de l'application du CR ResourceBackup que vous avez créé à l'étape 1.
   - **Spec.appVaultRef**: (*required*) Assurez-vous que cette valeur correspond à la référence appVaultRef de la CR ResourceBackup que vous avez créée à l'étape 1.
   - **Spec.appArchivePath** : Assurez-vous que cette valeur correspond à appArchivePath à partir du CR ResourceBackup que vous avez créé à l'étape 1.

     ```
     kubectl get backups <BACKUP_NAME> -n my-app-namespace -o
     jsonpath='{.status.appArchivePath}'
     ```

   - **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.stage**: (*required*) Une chaîne indiquant quelle étape de l'action doit être exécutée par le crochet d'exécution. Cette course de crochet ne fera pas courir de crochets dans une autre étape. Valeurs possibles :
     - Pré
     - Post

     Exemple YAML :

```
---
apiVersion: protect.trident.netapp.io/v1
kind: ExecHooksRun
metadata:
  name: example-hook-run
spec:
  applicationRef: my-app-name
  appVaultRef: my-appvault-name
  appArchivePath: example-resource-backup
  stage: Post
  action: Failover
```

3. Une fois que vous avez rempli le fichier CR avec les valeurs correctes, appliquez la CR :

```
kubectl apply -f trident-protect-hook-run.yaml
```

Utiliser l'interface de ligne de commande

Étapes

1. Créez la demande d'exécution manuelle du crochet :

```
tridentctl protect create exechooksrun <my_exec_hook_run_name>
-n <my_app_namespace> --action snapshot --stage <pre_or_post>
--app <my_app_name> --appvault <my_appvault_name> --path
<my_backup_name>
```

2. Vérifiez l'état de l'exécution du hook. Vous pouvez exécuter cette commande plusieurs fois jusqu'à ce que l'opération soit terminée :

```
tridentctl protect get exechooksrun -n <my_app_namespace>
<my_exec_hook_run_name>
```

3. Décrivez l'objet exechooksrun pour afficher les détails et l'état finaux :

```
kubectl -n <my_app_namespace> describe exechooksrun
<my_exec_hook_run_name>
```

Désinstallez Trident Protect

Vous devrez peut-être supprimer des composants de Trident Protect si vous passez d'une version d'essai à une version complète du produit.

Pour retirer Trident Protect, procédez comme suit.

Étapes

1. Supprimez les fichiers CR de Trident Protect :

   REMARQUE Cette étape n'est pas requise pour la version 25.06 et les versions ultérieures.

   ```
   helm uninstall -n trident-protect 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
   ```

Blogs de Trident et Trident Protect

Vous trouverez ici d'excellents blogs sur NetApp Trident et Trident Protect :

Blogs Trident

- 16 octobre 2025 :"Solutions de stockage avancées pour Kubernetes"
- 19 août 2025 :"Amélioration de la cohérence des données : instantanés de groupes de volumes dans la virtualisation OpenShift avec Trident"
- 9 mai 2025 : "Configuration automatique du backend Trident pour FSx pour ONTAP avec le module complémentaire Amazon EKS"
- 15 avril 2025 : "NetApp Trident avec Google Cloud NetApp Volumes pour le protocole SMB"
- 14 avril 2025 : "Exploiter le protocole Fibre Channel avec Trident 25.02 pour le stockage persistant sur Kubernetes"
- 14 avril 2025 : "Libérer la puissance des systèmes NetApp ASA r2 pour le stockage en mode bloc Kubernetes"
- 31 mars 2025 : "Simplification de l'installation de Trident sur Red Hat OpenShift avec le nouvel opérateur certifié"
- 27 mars 2025 : "Provisionnement de Trident pour PME avec Google Cloud NetApp Volumes"
- 5 mars 2025 : "Exploitez l'intégration transparente du stockage iSCSI : guide de FSxN sur ROSA clusters pour AWS"
- 27 février 2025 : "Déploiement de l'identité cloud avec Trident, GKE et Google Cloud NetApp Volumes"
- 12 décembre 2024 : "Présentation de la prise en charge de Fibre Channel dans Trident"
- 11 novembre 2024 : "NetApp Trident avec Google Cloud NetApp Volumes"
- 29 octobre 2024 : "Amazon FSx for NetApp ONTAP avec Red Hat OpenShift Service sur AWS (ROSA) à l'aide de Trident"
- 29 octobre 2024 : "Migration en direct de machines virtuelles avec OpenShift Virtualization sur ROSA et Amazon FSx for NetApp ONTAP"
- 8 juillet 2024 : "Utilisation de NVMe/TCP pour consommer le stockage ONTAP pour vos applications conteneurisées modernes sur Amazon EKS"
- 1er juillet 2024 : "Stockage Kubernetes transparent avec Google Cloud NetApp Volumes Flex et Astra Trident"
- 11 juin 2024 : "ONTAP comme stockage backend pour le registre d'images intégré dans OpenShift"

Blogs de Trident Protect

- 16 mai 2025 :"Automatisation du basculement du registre pour la reprise après sinistre avec les hooks post-restauration de Trident Protect"
- 16 mai 2025 : "Reprise après sinistre de la virtualisation OpenShift avec NetApp Trident Protect"
- 13 mai 2025 :"Migration de classes de stockage avec sauvegarde et restauration Trident Protect"
- 9 mai 2025 :"Redimensionnez les applications Kubernetes avec les hooks post-restauration de Trident Protect"
- 3 avril 2025 : "Trident Protect Power Up : réplication Kubernetes pour la protection et la reprise après

sinistre"

- 13 mars 2025 : "Opérations de sauvegarde et de restauration cohérentes après panne pour les machines virtuelles de virtualisation OpenShift"
- 11 mars 2025 : "Extension des modèles GitOps à la protection des données applicatives avec NetApp Trident"
- 03 mars 2025 : "Trident 25.02 : améliorer l'expérience Red Hat OpenShift avec de nouvelles fonctionnalités intéressantes"
- 15 janvier 2025 :"Présentation de Trident Protect, le contrôle d'accès basé sur les rôles"
- 11 novembre 2024 :"Gestion des données pilotée par Kubernetes : la nouvelle ère avec 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. Consultez "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 ?

L'installateur Trident crée un ClusterRole Kubernetes, qui dispose d'un accès spécifique aux ressources PersistentVolume, PersistentVolumeClaim, StorageClass et Secret du cluster Kubernetes. Reportez-vous à "Personnalisez l'installation tridentctl".

Puis-je générer localement les fichiers de manifeste exacts que Trident utilise pour l'installation ?

Vous pouvez générer et modifier localement les fichiers manifestes exacts utilisés par Trident pour l'installation, si nécessaire. Reportez-vous à "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. Cela permet de s'assurer que le même FlexVol volume 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.

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 la gestion et les DataLIF dans un fichier de définition ONTAP backend ?

Le LIF de gestion est obligatoire. DataLIF 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 : NetApp recommande 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, ce qui vous permet de créer un DNS circulaire pour équilibrer la charge sur plusieurs dataLIFs. 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 gestion et de données ?

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 le DataLIF sur le back-end ?

Vous pouvez mettre à jour DataLIF 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` pilote crée jusqu'à 200 qtrees dans le même FlexVol volume (configurable entre 50 et 300), 100,000 qtrees par nœud de cluster et 2,4 millions par cluster. Lorsque vous entrez un nouveau `PersistentVolumeClaim` qui est entretenu par le conducteur économique, le conducteur recherche s'il existe déjà un FlexVol volume qui peut réparer le nouveau qtree. Si le FlexVol volume n'existe pas et peut servir le Qtree, un nouveau FlexVol volume 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 ?

À compter d'aujourd'hui, la prise en charge des instantanés à la demande est disponible pour notre `ontap-nas` , `ontap-nas-flexgroup` , `ontap-san` , `ontap-san-economy` , `solidfire-san` , 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 volume, 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.

REMARQUE	Pour obtenir de l'aide sur Trident, créez un bundle de support à l'aide de `tridentctl logs -a -n trident` et envoyez-le au support NetApp.

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 les incluant `debugTraceFlags` dans votre définition back-end. Par exemple, inclure `debugTraceFlags: {"api":true, "method":true,}` pour obtenir des appels API et des traversées de méthode dans les journaux Trident. Les systèmes back-end existants peuvent avoir été `debugTraceFlags` configurés avec un `tridentctl backend update`.
- Lorsque vous utilisez Red Hat Enterprise Linux CoreOS (RHCOS), assurez-vous que `iscsid` est activé sur les nœuds worker et démarré par défaut. Pour ce faire, utilisez OpenShift MachineConfiguration ou modifiez les modèles d'allumage.
- Un problème courant que vous pourriez rencontrer lors de l'utilisation de Trident "Azure NetApp Files" est lorsque les secrets du locataire et du client proviennent d'un enregistrement d'application avec des autorisations insuffisantes. Pour une liste complète des exigences de Trident, consultez la "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.

AVERTISSEMENT	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.
```

Les clients NFSv4.2 signalent un « argument non valide » après la mise à niveau ONTAP alors qu'ils s'attendent à ce que « v4.2-xattrs » soit activé

Après la mise à niveau ONTAP, les clients NFSv4.2 peuvent signaler des erreurs « argument non valide » lors de la tentative de montage des exportations NFSv4.2. Ce problème se produit lorsque le `v4.2-xattrs` l'option n'est pas activée sur le SVM. .Solution de contournement Activer le `v4.2-xattrs` option sur le SVM ou mise à niveau vers ONTAP 9.12.1 ou version ultérieure, où cette option est activée par défaut.

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

"26,02"	Février 2026	Février 2027	Février 2028
"25,10"	Octobre 2026	Octobre 2027	Octobre 2028
"25,06"	Juin 2026	Juin 2027	Juin 2028
"25,02"	Février 2026	Février 2027	Février 2028
"24,10"	—	Octobre 2026	Octobre 2027
"24.06"	—	Juin 2026	Juin 2027
"24.02"	—	Février 2026	Février 2027
"23.10"	—	—	Octobre 2026
"23.07"	—	—	Juillet 2026
"23.04"	—	—	Avril 2026

Auto-assistance

Pour obtenir une liste complète des articles de dépannage, consultez "Base de connaissances NetApp (identifiant requis)".

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.

Présentation

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

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

Ports pour les nœuds de contrôleur

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

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

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

Ports pour les nœuds de travail

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

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

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

REMARQUE 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

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

Quand utiliser l'API REST

L'API REST est utile pour les installations avancées qui utilisent Trident 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.

REMARQUE	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.

AVERTISSEMENT	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`.

ASTUCE 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, ontap-san-economy
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
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
trident.netapp.io/skipRecoveryQueue	ignorer la file d'attente de récupération	ontap-nas, ontap-nas-économie, ontap-nas-flexgroup, ontap-san, ontap-san-économie

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 :

- NetApp recommande 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.

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

Lorsque vous créez une demande de volume persistant faisant référence à une configuration Trident `StorageClass`, Trident provisionne un nouveau volume en utilisant la classe de stockage correspondante et enregistre un nouveau volume persistant pour ce volume. Lors de la configuration du volume provisionné et du volume persistant correspondant, Trident respecte les règles suivantes :

- Trident génère un nom de volume persistant pour Kubernetes et un nom interne utilisé pour le provisionnement du stockage. Dans les deux cas, il garantit que les noms sont uniques dans leur périmètre.
- La taille du volume correspond le plus possible à la taille demandée dans le PVC, bien qu'elle puisse être arrondie à la quantité la plus proche, selon la plate-forme.

Kubernetes `StorageClass` objets

Kubernetes `StorageClass` les objets sont spécifiés par le nom dans `PersistentVolumeClaims` provisionner le stockage avec un ensemble de propriétés. La classe de stockage elle-même identifie le mécanisme de provisionnement à utiliser et définit cet ensemble de propriétés, comme le mécanisme de provisionnement le comprend.

Il s'agit de l'un des deux objets de base qui doivent être créés et gérés par l'administrateur. L'autre est l'objet back-end Trident.

Un Kubernetes `StorageClass` Voici quelques aspects d'un objet qui utilise Trident :

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: <Name>
provisioner: csi.trident.netapp.io
mountOptions: <Mount Options>
parameters: <Trident Parameters>
allowVolumeExpansion: true
volumeBindingMode: Immediate
```

Ces paramètres sont spécifiques à Trident et indiquent à Trident comment provisionner des volumes pour la classe.

Les paramètres de classe de stockage sont les suivants :

Attribut	Type	Obligatoire	Description
attributs	chaîne map[string]	non	Voir la section attributs ci-dessous
StoragePools	Mapper[string]StringList	non	Mappage des noms backend avec les listes de pools de stockage dans
Des médutiquesde stockage	Mapper[string]StringList	non	Mappage des noms backend avec les listes de pools de stockage dans
Exclus du stockagePools	Mapper[string]StringList	non	Mappage des noms backend avec les listes de pools de stockage dans

Les attributs de stockage et leurs valeurs possibles peuvent être classés en attributs de sélection des pools de stockage et en attributs Kubernetes.

Attributs de sélection du pool de stockage

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

Attribut	Type	Valeurs	Offre	Demande	Pris en charge par
support[1]	chaîne	hdd, hybride, ssd	Le pool contient des supports de ce type ; hybride signifie les deux	Type de support spécifié	ontap-nas, ontap-nas-économie, ontap-nas-flexgroup, ontap-san, solidfire-san

Attribut	Type	Valeurs	Offre	Demande	Pris en charge par
Type de provisionnement	chaîne	fin, épais	Le pool prend en charge cette méthode de provisionnement	Méthode de provisionnement spécifiée	thick : tous les systèmes ONTAP ; thin : tous les systèmes ONTAP et solidfire-san
Type de dos	chaîne	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, solidfire-san, 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
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
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, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, ontap-san-economy, solidfire-san, azure-netapp-files	1.11+

Attribut	Type	Valeurs	Description	Facteurs pertinents	Version Kubernetes
Volume Bindingmode	chaîne	Immédiat, WaitForFirstConsumer	Sélectionnez le moment où la liaison des volumes et le provisionnement dynamique se produisent	Tout	1.19 - 1.26

ASTUCE

- 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.

REMARQUE	Le cycle de vie d'un VolumeSnapshot est indépendant du PVC source : un snapshot persiste même après la suppression du PVC source. Lors de la suppression d'un 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.

REMARQUE	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.

Objets Kubernetes `VolumeGroupSnapshotClass`

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

UN `VolumeGroupSnapshotClass` La création d'un groupe d'instantanés doit être définie par un administrateur. Une classe d'instantanés de groupe de volumes est créée avec la définition suivante :

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

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

Objets Kubernetes `VolumeGroupSnapshot`

Un Kubernetes `VolumeGroupSnapshot` L'objet est une requête de création d'un instantané de plusieurs volumes. Tout comme un PVC représente une requête utilisateur pour un volume, un instantané de groupe de volumes est une requête utilisateur pour créer un instantané d'un PVC existant.

Lorsqu'une demande d'instantané de groupe de volumes arrive, Trident gère automatiquement la création de l'instantané de groupe pour les volumes sur le backend et expose l'instantané en créant un instantané unique. `VolumeGroupSnapshotContent` 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.

REMARQUE	Le cycle de vie d'un VolumeGroupSnapshot est indépendant du PVC source : un snapshot persiste même après la suppression du PVC source. Lors de la suppression d'un 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 snapshot du groupe de volumes est supprimé lorsque tous les snapshots associés sont supprimés.

Objets Kubernetes `VolumeGroupSnapshotContent`

Un Kubernetes `VolumeGroupSnapshotContent` l'objet représente un instantané de groupe pris à partir d'un volume déjà provisionné. Elle est similaire à un `PersistentVolume` et signifie un snapshot provisionné sur le cluster de stockage. Comme `PersistentVolumeClaim` pour les objets et `PersistentVolume`, lors de la création d'un Snapshot, l' `VolumeSnapshotContent`objet conserve un mappage un-à-un sur l' `VolumeSnapshot`objet qui avait demandé la création du Snapshot.

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

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

Kubernetes `CustomResourceDefinition` objets

Les ressources personnalisées Kubernetes sont des 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.

REMARQUE	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.

REMARQUE 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 backend. Pour ONTAP, cela correspond à des agrégats dans les SVM. Pour NetApp HCI/ SolidFire, cela correspond à des bandes QoS spécifiées par l'administrateur. Chaque pool de stockage possède un ensemble d'attributs de stockage distincts, qui définissent ses caractéristiques de performance et de protection des données.

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

ObjetsTrident `Volume`

Les volumes constituent l'unité de provisionnement de base, comprenant des terminaux back-end, tels que des partages NFS et des LUN iSCSI et FC. Dans Kubernetes, ces valeurs 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.

REMARQUE

- 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 »

Attribut	Type	Obligatoire	Description
Nom interne	chaîne	non	Nom de l'objet sur le système de stockage, généré par Trident
Volume cloneSourceVolume	chaîne	non	ONTAP (nas, san) et SolidFire-* : nom du volume à cloner
SplitOnClone	chaîne	non	ONTAP (nas, san) : séparer le clone de son parent
Politique de snapshots	chaîne	non	ONTAP-* : stratégie d'instantané à utiliser
Réserve de snapshots	chaîne	non	ONTAP-* : pourcentage de volume réservé pour les snapshots
ExportPolicy	chaîne	non	ontap-nas* : export policy à utiliser
Répertoire de snapshots	bool	non	ontap-nas* : indique si le répertoire des snapshots est visible
Autorisations unix	chaîne	non	ontap-nas* : autorisations UNIX initiales
Taille de bloc	chaîne	non	SolidFire-*: Taille de bloc/secteur
Système de fichiers	chaîne	non	Type de système de fichiers
ignorer la file d'attente de récupération	chaîne	non	Lors de la suppression d'un volume, ignorez la file d'attente de récupération dans le stockage et supprimez le volume immédiatement.

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

REMARQUE 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.

REMARQUE	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`

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.>`

AVERTISSEMENT	La modification des étiquettes de l'espace de noms peut entraîner l'absence de planification des modules, un "erreur de création: …" ou un "avertissement: trident-csi-…". Si cela se produit, vérifiez si le libellé de l'espace de noms pour `privileged` a été modifié. Si c'est le cas, réinstallez Trident.

Politiques de sécurité des pods (PSP)

Champ	Description	Valeur par défaut
`allowPrivilegeEscalation`	Les conteneurs privilégiés doivent autoriser l'escalade des privilèges.	`true`
`allowedCSIDrivers`	Trident n'utilise pas les volumes éphémères CSI en ligne.	Vide
`allowedCapabilities`	Les conteneurs Trident non privilégiés ne nécessitent pas de fonctionnalités supérieures à celles des ensembles par défaut et les conteneurs privilégiés se voient accorder toutes les capacités possibles.	Vide
`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`

Champ	Description	Valeur par défaut
forbiddenSysctls	Non sysctls sont autorisés.	Vide
fsGroup	Les conteneurs Trident s'exécutent en tant que root.	RunAsAny
hostIPC	Le montage des volumes NFS requiert la communication du IPC hôte avec nfsd	true
hostNetwork	Iscsiadm nécessite que le réseau hôte communique avec le démon iSCSI.	true
hostPID	Le PID hôte est requis pour vérifier si rpc-statd est en cours d'exécution sur le nœud.	true
hostPorts	Trident n'utilise aucun port hôte.	Vide
privileged	Les pods de nœuds Trident doivent exécuter un conteneur privilégié pour monter des volumes.	true
readOnlyRootFilesystem	Les pods de nœuds Trident doivent écrire dans le système de fichiers de nœuds.	false
requiredDropCapabilities	Les pods de nœuds Trident exécutent un conteneur privilégié et ne peuvent pas supprimer de fonctionnalités.	none
runAsGroup	Les conteneurs Trident s'exécutent en tant que root.	RunAsAny
runAsUser	Les conteneurs Trident s'exécutent en tant que root.	runAsAny
runtimeClass	Trident n'utilise pas RuntimeClasses.	Vide
seLinux	Trident n'est pas défini seLinuxOptions Car il existe actuellement des différences dans le mode de gestion des conteneurs et de distribution Kubernetes de SELinux.	Vide
supplementalGroups	Les conteneurs Trident s'exécutent en tant que root.	RunAsAny
volumes	Les pods Trident requièrent ces plug-ins de volume.	hostPath, projected, emptyDir

Contraintes de contexte de sécurité (SCC)

Étiquettes	Description	Valeur par défaut
`allowHostDirVolumePlugin`	Les pods des nœuds Trident montent le système de fichiers racine du nœud.	`true`
`allowHostIPC`	Le montage des volumes NFS requiert la communication du IPC hôte avec `nfsd`.	`true`
`allowHostNetwork`	Iscsiadm nécessite que le réseau hôte communique avec le démon iSCSI.	`true`
`allowHostPID`	Le PID hôte est requis pour vérifier si `rpc-statd` est en cours d'exécution sur le nœud.	`true`
`allowHostPorts`	Trident n'utilise aucun port hôte.	`false`
`allowPrivilegeEscalation`	Les conteneurs privilégiés doivent autoriser l'escalade des privilèges.	`true`
`allowPrivilegedContainer`	Les pods de nœuds Trident doivent exécuter un conteneur privilégié pour monter des volumes.	`true`
`allowedUnsafeSysctls`	Trident n'exige aucun niveau de sécurité `sysctls`.	`none`
`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`

Étiquettes	Description	Valeur par défaut
`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

Copyright © 2026 NetApp, Inc. Tous droits réservés. Imprimé aux États-Unis. Aucune partie de ce document protégé par copyright ne peut être reproduite sous quelque forme que ce soit ou selon quelque méthode que ce soit (graphique, électronique ou mécanique, notamment par photocopie, enregistrement ou stockage dans un système de récupération électronique) sans l'autorisation écrite préalable du détenteur du droit de copyright.

Les logiciels dérivés des éléments NetApp protégés par copyright sont soumis à la licence et à l'avis de non-responsabilité suivants :

CE LOGICIEL EST FOURNI PAR NETAPP « EN L'ÉTAT » ET SANS GARANTIES EXPRESSES OU TACITES, Y COMPRIS LES GARANTIES TACITES DE QUALITÉ MARCHANDE ET D'ADÉQUATION À UN USAGE PARTICULIER, QUI SONT EXCLUES PAR LES PRÉSENTES. EN AUCUN CAS NETAPP NE SERA TENU POUR RESPONSABLE DE DOMMAGES DIRECTS, INDIRECTS, ACCESSOIRES, PARTICULIERS OU EXEMPLAIRES (Y COMPRIS L'ACHAT DE BIENS ET DE SERVICES DE SUBSTITUTION, LA PERTE DE JOUISSANCE, DE DONNÉES OU DE PROFITS, OU L'INTERRUPTION D'ACTIVITÉ), QUELLES QU'EN SOIENT LA CAUSE ET LA DOCTRINE DE RESPONSABILITÉ, QU'IL S'AGISSE DE RESPONSABILITÉ CONTRACTUELLE, STRICTE OU DÉLICTUELLE (Y COMPRIS LA NÉGLIGENCE OU AUTRE) DÉCOULANT DE L'UTILISATION DE CE LOGICIEL, MÊME SI LA SOCIÉTÉ A ÉTÉ INFORMÉE DE LA POSSIBILITÉ DE TELS DOMMAGES.

NetApp se réserve le droit de modifier les produits décrits dans le présent document à tout moment et sans préavis. NetApp décline toute responsabilité découlant de l'utilisation des produits décrits dans le présent document, sauf accord explicite écrit de NetApp. L'utilisation ou l'achat de ce produit ne concède pas de licence dans le cadre de droits de brevet, de droits de marque commerciale ou de tout autre droit de propriété intellectuelle de NetApp.

Le produit décrit dans ce manuel peut être protégé par un ou plusieurs brevets américains, étrangers ou par une demande en attente.

LÉGENDE DE RESTRICTION DES DROITS : L'utilisation, la duplication ou la divulgation par le gouvernement sont sujettes aux restrictions énoncées dans le sous-paragraphe (b)(3) de la clause Rights in Technical Data-Noncommercial Items du DFARS 252.227-7013 (février 2014) et du FAR 52.227-19 (décembre 2007).

Les données contenues dans les présentes se rapportent à un produit et/ou service commercial (tel que défini par la clause FAR 2.101). Il s'agit de données propriétaires de NetApp, Inc. Toutes les données techniques et tous les logiciels fournis par NetApp en vertu du présent Accord sont à caractère commercial et ont été exclusivement développés à l'aide de fonds privés. Le gouvernement des États-Unis dispose d'une licence limitée irrévocable, non exclusive, non cessible, non transférable et mondiale. Cette licence lui permet d'utiliser uniquement les données relatives au contrat du gouvernement des États-Unis d'après lequel les données lui ont été fournies ou celles qui sont nécessaires à son exécution. Sauf dispositions contraires énoncées dans les présentes, l'utilisation, la divulgation, la reproduction, la modification, l'exécution, l'affichage des données sont interdits sans avoir obtenu le consentement écrit préalable de NetApp, Inc. Les droits de licences du Département de la Défense du gouvernement des États-Unis se limitent aux droits identifiés par la clause 252.227-7015(b) du DFARS (février 2014).

Informations sur les marques commerciales

NETAPP, le logo NETAPP et les marques citées sur le site http://www.netapp.com/TM sont des marques déposées ou des marques commerciales de NetApp, Inc. Les autres noms de marques et de produits sont des marques commerciales de leurs propriétaires respectifs.