Documentation Trident 26.02

Trident

NetApp
August 04, 2026

This PDF was generated from https://docs.netapp.com/fr-fr/trident-2602/index.html on August 04, 2026.
Always check docs.netapp.com for the latest.

Sommaire

Documentation Trident 26.02 . . . 1
Notes de version . . . 2
Quoi de neuf . . . 2
Nouveautés de la version 26.02.1 . . . 2
Quoi de neuf dans la version 26.02 . . . 3
Quoi de neuf dans la version 25.10 . . . 6
Modifications dans la version 25.06.2 . . . 8
Modifications dans la version 25.06.1 . . . 8
Modifications dans la version 25.06 . . . 8
Modifications dans la version 25.02.1 . . . 10
Modifications dans la version 25.02 . . . 11
Modifications dans la version 24.10.1 . . . 13
Modifications dans la version 24.10 . . . 13
Modifications dans la version 24.06 . . . 15
Modifications dans la version 24.02 . . . 15
Changements dans la version 23.10 . . . 16
Modifications dans 23.07.1 . . . 17
Modifications dans 23.07 . . . 17
Modifications dans la version 23.04 . . . 18
Modifications dans 23.01.1 . . . 19
Modifications dans 23.01 . . . 19
Changements dans 22.10 . . . 20
Modifications dans la version 22.07 . . . 21
Modifications dans 22.04 . . . 22
Modifications dans 22.01.1 . . . 23
Modifications dans la version 22.01.0 . . . 23
Modifications dans 21.10.1 . . . 24
Modifications dans la version 21.10.0 . . . 24
Problèmes connus . . . 25
Pour plus d'informations . . . 26
Versions antérieures de la documentation . . . 26
NetApp Trident prise en charge de Trident pour les systèmes de stockage ONTAP ASA r2 . . . 27
Opérations prises en charge . . . 27
Opérations non prises en charge . . . 27
Problèmes connus . . . 28
VolumeSnapshots n'atteignent pas l'état ReadyToUse . . . 28
La restauration des sauvegardes Restic de fichiers volumineux peut échouer . . . 28
Commencer . . . 29
En savoir plus sur Trident . . . 29
En savoir plus sur Trident . . . 29
Architecture Trident . . . 30
Concepts . . . 33
Démarrage rapide pour Trident . . . 37

Et ensuite ? . . . 38
Exigences . . . 38
Informations essentielles sur Trident . . . 38
Interfaces prises en charge (orchestrators) . . . 39
Backends pris en charge (stockage) . . . 39
Prise en charge de Trident pour KubeVirt et la virtualisation OpenShift . . . 39
Exigences fonctionnelles . . . 40
Systèmes d'exploitation hôtes testés . . . 40
Configuration de l'hôte . . . 41
Configuration du système de stockage . . . 41
Ports Trident . . . 41
Images de conteneur et versions Kubernetes correspondantes . . . 41
Installer Trident . . . 43
Découvrez l'installation de Trident . . . 43
Informations essentielles concernant Trident 26.02 . . . 43
Avant de commencer . . . 43
Choisissez votre méthode d'installation . . . 44
Choisissez votre mode d'installation . . . 46
Sélectionnez le processus en fonction de votre méthode et de votre mode . . . 46
Passer d'une méthode d'installation à une autre . . . 47
Autres options de configuration connues . . . 47
Installer à l'aide de l'opérateur Trident . . . 47
Déployez manuellement l'opérateur Trident (mode standard) . . . 47
Déployez manuellement l'opérateur Trident (mode hors ligne) . . . 53
Déployez l'opérateur Trident à l'aide de Helm (mode standard) . . . 59
Déployez l'opérateur Trident à l'aide de Helm (mode hors ligne) . . . 67
Personnaliser l'installation de l'opérateur Trident . . . 75
Installer à l'aide de tridentctl . . . 87
Installer à l'aide de tridentctl . . . 87
Personnaliser l'installation de tridentctl . . . 91
Installer à l'aide de l'opérateur certifié OpenShift . . . 92
Installez Trident en utilisant OpenShift OperatorHub . . . 92
Passez à l'opérateur Trident certifié OpenShift . . . 95
Utilisez Trident . . . 99
Préparez le nœud de travail . . . 99
Choisir les bons outils . . . 99
Découverte de service de nœud . . . 99
Volumes NFS . . . 100
volumes iSCSI . . . 100
Volumes NVMe/TCP . . . 104
Volumes SCSI sur FC . . . 105
Préparez-vous à provisionner des volumes SMB . . . 108
Configurer et gérer les backends . . . 109
Configurer les backends . . . 109
Azure NetApp Files . . . 110

Google Cloud NetApp Volumes . . . 130
Configurer un backend NetApp HCI ou SolidFire . . . 157
Pilotes ONTAP SAN . . . 163
Pilotes ONTAP NAS . . . 193
Amazon FSx for NetApp ONTAP . . . 232
Créer des backends avec kubectl . . . 271
Gérer les backends . . . 278
Créer et gérer des classes de stockage . . . 288
Créer une classe de stockage . . . 288
Gérer les classes de stockage . . . 291
Provisionner et gérer les volumes . . . 293
Provisionner un volume . . . 293
Étendre les volumes . . . 297
Comprendre les limites du sous-système RWX NVMe . . . 308
Évolutivité du contrôleur . . . 310
Extension automatique du volume . . . 314
Gérer les stratégies Autogrow . . . 320
Importer des volumes . . . 330
Personnalisez les noms et les étiquettes des volumes . . . 342
Partager un volume NFS entre espaces de noms . . . 345
Cloner des volumes entre espaces de noms . . . 349
Répliquer les volumes à l'aide de SnapMirror . . . 352
Utiliser la topologie CSI . . . 358
Travailler avec des snapshots . . . 366
Travailler avec les instantanés de groupe de volumes . . . 374
Gérer et surveiller Trident . . . 379
Mettre à niveau Trident . . . 379
Mettre à niveau Trident . . . 379
Mise à niveau avec l'opérateur . . . 380
Mise à niveau avec tridentctl . . . 385
Gérez Trident à l'aide de tridentctl . . . 386
Commandes et drapeaux globaux . . . 386
Options de commande et drapeaux . . . 388
Prise en charge des plugins . . . 394
Surveiller Trident . . . 394
Aperçu . . . 394
Étape 1 : Définir une cible Prometheus . . . 395
Étape 2 : Créer un ServiceMonitor Prometheus . . . 395
Étape 3 : Interroger les métriques Trident avec PromQL . . . 397
Découvrez la télémétrie AutoSupport de Trident . . . 398
Désactiver les métriques Trident . . . 399
Désinstaller Trident . . . 399
Déterminez la méthode d'installation d'origine . . . 399
Désinstaller une installation de l'opérateur Trident . . . 399
Désinstaller une `tridentctl` installation . . . 400

Trident for Docker . . . 401
Prérequis pour le déploiement . . . 401
Vérifiez les exigences . . . 401
Outils NVMe . . . 403
Outils FC . . . 404
Déployer Trident . . . 406
Méthode de plugin géré par Docker (version 1.13/17.03 et ultérieure) . . . 406
Méthode traditionnelle (version 1.12 ou antérieure) . . . 408
Démarrer Trident au démarrage du système . . . 409
Mettre à niveau ou désinstaller Trident . . . 410
Mise à niveau . . . 411
Désinstaller . . . 412
Travailler avec des volumes . . . 412
Créer un volume . . . 412
Supprimer un volume . . . 413
Cloner un volume . . . 413
Accéder aux volumes créés en externe . . . 415
Options de volume spécifiques au pilote . . . 415
Collecter les journaux . . . 420
Collecter les journaux pour le dépannage . . . 420
Conseils généraux de dépannage . . . 421
Gérer plusieurs instances Trident . . . 421
Étapes pour le plugin géré par Docker (version 1.13/17.03 ou ultérieure) . . . 421
Étapes pour la version traditionnelle (1.12 ou antérieure) . . . 422
Options de configuration du stockage . . . 422
Options de configuration globales . . . 423
Configuration ONTAP . . . 423
Configuration du logiciel Element . . . 432
Problèmes et limitations connus . . . 434
La mise à niveau du plugin Trident Docker Volume vers la version 20.10 et ultérieures à partir de versions plus anciennes entraîne un échec de mise à niveau avec l'erreur « no such file or directory ». . . . 434
Les noms de volumes doivent comporter au minimum 2 caractères. . . . 435
Docker Swarm présente certains comportements qui empêchent Trident de le prendre en charge avec chaque combinaison de stockage et de pilote. . . . 435
Si un FlexGroup est en cours de provisionnement, ONTAP ne provisionne pas un second FlexGroup si le second FlexGroup a un ou plusieurs agrégats en commun avec le FlexGroup en cours de provisionnement. . . . 435
Meilleures pratiques et recommandations . . . 436
Déploiement . . . 436
Déployer dans un espace de noms dédié . . . 436
Utilisez des quotas et des limites de plage pour contrôler la consommation de stockage . . . 436
Configuration du stockage . . . 436
Présentation de la plateforme . . . 436
Meilleures pratiques ONTAP et Cloud Volumes ONTAP . . . 437
SolidFire meilleures pratiques . . . 441

Où trouver plus d'informations ? . . . 443
Intégrer Trident . . . 443
Sélection et déploiement du pilote . . . 444
Conception de classe de stockage . . . 446
Conception de pool virtuel . . . 447
Opérations de volume . . . 448
Service de métriques . . . 452
Protection des données et reprise après sinistre . . . 453
Réplication et récupération de Trident . . . 453
Réplication et récupération de SVM . . . 454
Réplication et récupération de volume . . . 455
protection des données Snapshot . . . 455
Automatisation du basculement des applications avec état avec Trident . . . 455
Détails concernant le détachement forcé . . . 455
Détails sur le basculement automatique . . . 456
Sécurité . . . 461
Sécurité . . . 461
Linux Unified Key Setup (LUKS) . . . 462
Chiffrement Kerberos en vol . . . 469
Protégez les applications avec Trident Protect . . . 477
Découvrez Trident Protect . . . 477
Et ensuite ? . . . 477
Installer Trident Protect . . . 477
Exigences de Trident Protect . . . 477
Installez et configurez Trident Protect . . . 481
Installez le plugin CLI Trident Protect . . . 485
Personnaliser l'installation de Trident Protect . . . 489
Gérer Trident Protect . . . 494
Gérer l'autorisation Trident et le contrôle d'accès . . . 494
Surveiller les ressources de Trident Protect . . . 501
Générer un bundle de support Trident Protect . . . 506
Mettre à niveau Trident Protect . . . 508
Gérer et protéger les applications . . . 510
Utilisez les objets Trident Protect AppVault pour gérer les compartiments . . . 510
Définir une application pour la gestion avec Trident Protect . . . 524
Protégez les applications à l'aide de Trident Protect . . . 529
Restaurer les applications . . . 542
Répliquez les applications à l'aide de NetApp SnapMirror et Trident Protect . . . 560
Migrer les applications à l'aide de Trident Protect . . . 577
Gérer les hooks d'exécution de Trident Protect . . . 581
Désinstaller Trident Protect . . . 593
Blogs de Trident et Trident Protect . . . 594
Blogs de Trident . . . 594
Blogs de Trident Protect . . . 594
Connaissances et support . . . 596

Foire aux questions . . . 596
Questions générales . . . 596
Installer et utiliser Trident sur un cluster Kubernetes . . . 596
Dépannage et support . . . 597
Mettre à niveau Trident . . . 598
Gérer les backends et les volumes . . . 599
Dépannage . . . 603
Dépannage général . . . 603
Déploiement Trident infructueux avec l'opérateur . . . 604
Déploiement Trident infructueux utilisant `tridentctl` . . . 606
Supprimez complètement Trident et les CRD . . . 607
Échec du déstockage du nœud NVMe avec les espaces de noms de blocs bruts RWX sous Kubernetes 1.26 . . . 607
Les clients NFSv4.2 signalent « invalid argument » après la mise à niveau ONTAP alors qu'ils s'attendent à ce que « v4.2-xattrs » soit activé . . . 608
Support . . . 608
Cycle de vie du support Trident . . . 608
Autonomie . . . 609
soutien communautaire . . . 609
Support technique NetApp . . . 609
Pour plus d'informations . . . 609
Référence . . . 610
Ports Trident . . . 610
Aperçu . . . 610
API REST Trident . . . 612
Quand utiliser l'API REST . . . 612
Utilisation de l'API REST . . . 613
Options de ligne de commande . . . 613
Journalisation . . . 613
Kubernetes . . . 614
Docker . . . 614
REST . . . 614
Objets Kubernetes et Trident . . . 615
Comment les objets interagissent-ils entre eux ? . . . 615
Objets `PersistentVolumeClaim` Kubernetes . . . 615
Objets `PersistentVolume` Kubernetes . . . 617
Objets `StorageClass` Kubernetes . . . 617
Objets `VolumeSnapshotClass` Kubernetes . . . 621
Objets `VolumeSnapshot` Kubernetes . . . 621
Objets `VolumeSnapshotContent` Kubernetes . . . 622
Objets `VolumeGroupSnapshotClass` Kubernetes . . . 622
Objets `VolumeGroupSnapshot` Kubernetes . . . 623
Objets `VolumeGroupSnapshotContent` Kubernetes . . . 623
Objets `CustomResourceDefinition` Kubernetes . . . 624

Trident `StorageClass` objets . . . 624
Objets backend Trident . . . 624
Trident `StoragePool` objets . . . 625
Trident `Volume` objets . . . 625
Trident `Snapshot` objets . . . 626
Trident `ResourceQuota` objet . . . 627
Normes de sécurité des pods (PSS) et Security Context Constraints (SCC) . . . 628
Contexte de sécurité Kubernetes requis et champs associés . . . 629
Normes de sécurité des pods (PSS) . . . 629
Politiques de sécurité des pods (PSP) . . . 630
Security Context Constraints (SCC) . . . 631
Mentions légales . . . 634
Copyright . . . 634
Marques déposées . . . 634
Brevets . . . 634
Politique de confidentialité . . . 634
Open source . . . 634

Documentation Trident 26.02

Notes de version

Quoi de neuf

Les notes de version fournissent des informations sur les nouvelles fonctionnalités, les améliorations et les corrections de bugs dans la dernière version de NetApp Trident.

> Le `tridentctl` binaire pour Linux fourni dans le fichier zip d'installation est la version testée et prise en charge. Veuillez noter que le `macos` binaire fourni dans la `/extras` partie du fichier zip n'est ni testé ni pris en charge.

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.

Trident Protect

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 le "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 le "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 le "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 n°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".

Trident Protect

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 "Créer 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, y compris les améliorations, les correctifs et les suppressions.

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 d'ONTAP-SAN (iSCSI et FC). Voir "Travailler avec les instantanés de groupe de volumes".
  - Ajout de la prise en charge du basculement automatique des charges de travail avec détachement forcé des volumes pour 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 avec 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 d'ONTAP AFX pour le pilote ONTAP NAS. Voir "Options et exemples de configuration 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 TridentOrchestrator CR et les valeurs du Helm chart. ("Problème n°1000", "Problème n°927", "Problème n°853", "Problème n°592", "Problème n°110").
  - Ajout de la prise en charge FC pour la personnalité ASAr2. Voir "Options et exemples de configuration SAN ONTAP".
  - Ajout d'une option permettant de diffuser les métriques Prometheus via HTTPS au lieu de HTTP. Voir "Surveiller Trident".
  - Ajout d'une option `--no-rename` lors de l'importation d'un volume permettant de conserver le nom d'origine tout en laissant 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, operator et tridentctl ("Problème n°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 unified ONTAP 9), en plus de l'aperçu technique existant pour le pilote ONTAP-SAN (protocoles iSCSI et FCP dans unified ONTAP 9).

Correctifs

- **Kubernetes** :

- Correction de l'incohérence du nom du conteneur CSI node-driver-registrar en normalisant le DaemonSet Linux en node-driver-registrar pour correspondre au DaemonSet Windows et à la dénomination des images de conteneur.
- Correction d'un problème où les politiques d'exportation pour les qtrees hérités n'étaient pas correctement mises à niveau.

- **Openshift** :
  - Correction du problème de démarrage du pod de nœud Trident sur les nœuds Windows dans Openshift en raison de SCC ayant allowHostDirVolumePlugin défini sur false ("Problème n°950").
- Correction d'un problème où le QPS de l'API Kubernetes n'était pas défini via Helm ("Problème n°975").
- Correction de l'impossibilité de monter une revendication de volume persistant (PVC) basée sur un instantané d'un PVC de système de fichiers XFS basé sur NVMe sur le même nœud Kubernetes.
- Correction du 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 24.10 et supérieures, résolvant le problème « invalid SANType ».
- Correction d'un problème où l'état du backend Trident ne passait pas à en ligne/hors ligne sans redémarrer le contrôleur Trident.
- Correction d'un problème de concurrence intermittent provoquant un redimensionnement lent des PVC.
- Correction des instantanés qui n'étaient pas supprimés en cas d'échec de clonage de volume.
- Correction de l'échec de la désactivation du volume lorsque son chemin d'accès au périphérique était modifié par le noyau.
- Correction d'un échec de désinstallation du volume dû à un périphérique LUKS déjà fermé.
- Correction d'un problème où la lenteur des opérations de stockage entraînait des erreurs ContextDeadline.
- L'opérateur Trident attendra le délai d'expiration k8s configurable pour vérifier la version de Trident.

Trident Protect

NetApp Trident Protect offre des capacités avancées de gestion des données applicatives qui améliorent la fonctionnalité et la disponibilité des applications Kubernetes avec état, soutenues 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 de planification 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 "Annotations de sauvegarde et de planification prises en charge".
- Ajout d'une annotation à la CR de planification 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 "Annotations de sauvegarde et de planification prises en charge".
- Amélioration `tridentctl-protect` des listes de sauvegardes et d'instantanés avec un nouveau champ pour indiquer les échecs des hooks d'exécution.

Modifications dans la version 25.06.2

Trident

Correctifs

- **Kubernetes** : Correction d'un problème critique 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 dissociés des sous-systèmes.
  - Correction d'un problème où plusieurs tentatives de fermeture d'un périphérique LUKS entraînaient des échecs de détachement des volumes.
  - Correction du déblocage du volume iSCSI lorsque le chemin d'accès au périphérique a changé depuis sa création.
  - Clonage par blocs de volumes entre classes de stockage.
- **OpenShift**: Correction d'un problème où la préparation du nœud iSCSI échouait avec OCP 4.19.
- Augmentation du délai d'attente lors du clonage d'un volume à l'aide de backends SolidFire ("Problème n°1008").

Modifications dans la version 25.06

Trident

Améliorations

- **Kubernetes** :
  - Ajout de la prise en charge des instantanés de groupes de volumes CSI avec `v1beta1` Volume Group Snapshot Kubernetes APIs pour le pilote iSCSI ONTAP-SAN. Voir "Travailler avec les instantanés de groupe 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 SAN ONTAP".
  - Ajout de la prise en charge sécurisée du protocole SMB 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 des nœuds Trident améliorée pour une meilleure évolutivité des opérations sur les nœuds pour les volumes iSCSI.
- Ajouté `--allow-discards` lors de l'ouverture des volumes LUKS pour permettre 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 ayant échoué mais partiellement formatés.
- Idempotence améliorée des nœuds Trident pour l'attachement 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 volumes avec SnapMirror pour les backends NVMe. Voir "Répliquer les volumes à l'aide de SnapMirror".

Améliorations expérimentales

> Ne pas utiliser dans des environnements de production.

- [Aperçu technique] Activation des opérations simultanées du contrôleur Trident via le `--enable-concurrency` feature flag. Cela permet aux opérations du contrôleur de s'exécuter en parallèle, améliorant les performances pour les environnements occupés 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 de QOS avec le pilote ANF.

Correctifs

- **Kubernetes** :
  - Correction d'un problème lié à CSI NodeExpandVolume où les périphériques multipath pouvaient présenter des tailles incohérentes lorsque le ou les disques SCSI sous-jacents étaient indisponibles.
  - Correction de l'échec du nettoyage des règles d'export 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.
  - Correction d'un problème de déploiement lors de l'installation de Trident à l'aide de Kustomize ("Problème n° 831").
  - Correction des règles d'export manquantes pour les PVC créés à partir de snapshots ("Problème n°1016").
  - Correction d'un problème 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.
- Correction d'un problème lié à l'extension des volumes ONTAP-NAS-Economy jusqu'à 300 To malgré des échecs de redimensionnement.
- Correction d'un problème où les opérations de division de clones étaient effectuées de manière synchrone lors de l'utilisation de l'API REST ONTAP.

Dépréciations :

- **Kubernetes** : version minimale de Kubernetes prise en charge mise à jour à v1.27.

Trident Protect

NetApp Trident Protect offre des capacités avancées de gestion des données applicatives qui améliorent la fonctionnalité et la disponibilité des applications Kubernetes avec état, soutenues 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.
- Amélioration de la granularité de la définition des applications et de la restauration sélective grâce au filtrage Group-Version-Kind (GVK).
- Resynchronisation et réplication inverse efficaces lors de l'utilisation de AppMirrorRelationship (AMR) avec NetApp SnapMirror, pour éviter la réplication complète du PVC.
- Ajout de la possibilité d'utiliser EKS Pod Identity 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) va maintenant vérifier l'expansion du PVC source et effectuer l'expansion appropriée sur le PVC de destination si nécessaire.

Correctifs

- Correction d'un bug où les valeurs d'annotation des instantanés précédents étaient appliquées aux nouveaux instantanés. Toutes les annotations des instantanés sont désormais appliquées correctement.
- Un secret a été défini pour le chiffrement du transporteur de données (Kopia / Restic) par défaut, s'il n'est pas défini.
- Ajout d'une validation améliorée et de messages d'erreur améliorés pour la création de S3 appvault.
- AppMirrorRelationship (AMR) ne réplique désormais que les PV dans l'état Bound, afin d'éviter les tentatives infructueuses.
- Correction d'un problème où des erreurs étaient affichées lors de l'obtention de AppVaultContent sur un AppVault comportant un grand nombre de sauvegardes.
- KubeVirt VMSnapshots sont exclus des opérations de restauration et de basculement afin d'éviter les échecs.
- Correction d'un problème avec Kopia où les instantanés étaient supprimés prématurément en raison du calendrier de rétention par défaut de Kopia qui remplaçait celui défini par l'utilisateur dans le calendrier.

Modifications dans la version 25.02.1

Trident

Correctifs

- **Kubernetes** :

- Correction d'un problème dans l'opérateur trident où les noms et versions des images sidecar étaient incorrectement renseignés lors de l'utilisation d'un registre d'images non par défaut ("Problème n° 983").
- Correction du problème où les sessions multipath ne parviennent pas à se rétablir lors d'un retour de basculement ONTAP ("Problème n°961").

Modifications dans la version 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** :
  - Ajout de la prise en charge de ONTAP ASA r2 pour iSCSI.
  - Prise en charge du détachement forcé pour les volumes ONTAP-NAS lors de scénarios d'arrêt de nœud non progressif. Les nouveaux volumes ONTAP-NAS utiliseront désormais des règles d'export par volume gérées par Trident. Un chemin de mise à niveau est fourni pour permettre aux volumes existants de passer au nouveau modèle de règles d'export lors de la dépublication, sans affecter les charges de travail actives.
  - Ajout de l'annotation cloneFromSnapshot.
  - Ajout de la prise en charge du clonage de volumes entre espaces de noms.
  - Amélioration des corrections d'analyse d'auto-réparation iSCSI pour lancer de nouvelles analyses par hôte, canal, cible et ID LUN exacts.
  - Ajout de la 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 OpenShift Virtualization pour les pilotes ONTAP.
- Ajout de la prise en charge de Fibre Channel sur le pilote ONTAP-SAN.
- Ajout de la prise en charge de NVMe LUKS.
- Passage à l'image scratch pour toutes les images de base.
- Ajout de la découverte et de la journalisation de l'état de connexion iSCSI lorsque les sessions iSCSI devraient être connectées mais ne le sont pas ("Problème n°961").
- Ajout de la prise en charge des volumes SMB avec le driver google-cloud-netapp-volumes.
- Ajout de la prise en charge permettant aux volumes ONTAP d'ignorer la file d'attente de récupération lors de leur suppression.
- Ajout de la prise en charge du remplacement des images par défaut à l'aide de SHA au lieu de balises.
- Ajout du paramètre image-pull-secrets à l'installateur tridentctl.

Correctifs

- **Kubernetes** :

- Correction des adresses IP de nœuds manquantes dans les règles d'export automatique ("Problème n°965").
- Correction du passage prématuré des règles d'export automatiques à une règle par volume pour ONTAP-NAS-Economy.
- Correction des informations d'identification de la configuration du backend pour prendre en charge toutes les partitions AWS ARN disponibles ("Problème n°913").
- Ajout d'une option pour désactiver la réconciliation automatique du configurateur dans l'opérateur Trident ("Problème n°924").
- Ajouté securityContext pour le conteneur csi-resizer ("Problème n°976").

Trident Protect

NetApp Trident Protect offre des capacités avancées de gestion des données applicatives qui améliorent la fonctionnalité et la disponibilité des applications Kubernetes avec état, soutenues 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 KubeVirt / OpenShift Virtualization VMs pour volumeMode : File et volumeMode : Block (stockage de 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 possibilité de contrôler le comportement de gel au niveau de l'application pour les environnements Kubevirt.
- Ajout de la prise en charge de la configuration des connexions proxy AutoSupport.
- Ajout de la possibilité de définir un secret pour le chiffrement des data mover (Kopia / Restic).
- Ajout de la possibilité d'exécuter manuellement un hook 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.
- Extension de ResourceFilter pour permettre l'exclusion des ressources à l'échelle du cluster.
- Ajout de la prise en charge du jeton de session AWS dans les informations d'identification S3 AppVault.
- Ajout de la prise en charge de la collecte des ressources après les hooks d'exécution pré-snapshot.

Correctifs

- Amélioration de la gestion des volumes temporaires pour éviter la file d'attente de récupération des volumes ONTAP.
- Les annotations SCC sont maintenant restaurées à leurs valeurs d'origine.
- Amélioration de l'efficacité de la restauration grâce à la prise en charge des opérations parallèles.
- Prise en charge améliorée des délais d'expiration des hooks d'exécution pour les applications plus volumineuses.

Modifications dans la version 24.10.1

Améliorations

- **Kubernetes** : prise en charge de Kubernetes 1.32 ajoutée.
- Ajout de la découverte et de la journalisation de l'état de connexion iSCSI lorsque les sessions iSCSI devraient être connectées mais ne le sont pas ("Problème n°961").

Correctifs

- Correction des adresses IP de nœuds manquantes dans les règles d'export automatique ("Problème n°965").
- Correction du passage prématuré des règles d'export automatiques à une règle par volume pour ONTAP-NAS-Economy.
- Mise à jour des dépendances de Trident et Trident-ASUP pour corriger CVE-2024-45337 et CVE-2024-45310.
- Suppression des déconnexions pour les portails non-CHAP présentant des dysfonctionnements intermittents pendant l'auto-réparation iSCSI ("Problème n°961").

Modifications dans la version 24.10

Améliorations

- Le pilote Google Cloud NetApp Volumes est désormais disponible de manière générale pour les volumes NFS et prend en charge le provisionnement tenant compte des zones.
- GCP Workload Identity sera utilisée comme identité cloud pour Google Cloud NetApp Volumes avec GKE.
- Ajout `formatOptions` d'un paramètre de configuration aux pilotes ONTAP-SAN et ONTAP-SAN-Economy pour permettre aux utilisateurs de spécifier les options de format LUN.
- Taille minimale des volumes Azure NetApp Files réduite à 50 Gio. La nouvelle taille minimale d'Azure devrait être généralement disponible en novembre.
- Ajout `denyNewVolumePools` d'un paramètre de configuration pour restreindre les pilotes ONTAP-NAS-Economy et ONTAP-SAN-Economy aux pools FlexVol® préexistants.
- Ajout d'une détection pour l'ajout, la suppression ou le renommage d'agrégats du SVM sur tous les pilotes ONTAP.
- Ajout de 18 MiB de surcharge aux LUNs LUKS pour garantir que la taille de PVC signalée soit utilisable.
- Amélioration de la gestion des erreurs de mise en scène et de désinstallation des nœuds ONTAP-SAN et ONTAP-SAN-Economy afin de permettre la désinstallation de périphériques après un échec de mise en scène.
- 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` ("Problème n°792").

Kubernetes

- Ajout de nouvelles fonctionnalités Trident pour les flux de travail natifs de Kubernetes :
  - Protection des données
  - Migration de données

- Reprise après sinistre
- Mobilité des applications

  "En savoir plus sur Trident Protect".

- Ajout d'une nouvelle option `--k8s-api-qps` aux programmes d'installation pour définir la valeur QPS utilisée par Trident pour communiquer avec le serveur d'API Kubernetes.
- Ajouté `--node-prep` un indicateur aux installateurs pour la gestion automatique des dépendances du protocole de stockage sur les nœuds du 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 lors de scénarios d'arrêt de nœud non progressif.
- Les nouveaux volumes NFS ONTAP-NAS-Economy utiliseront des règles d'export par qtree lors de l'utilisation de l'option `autoExportPolicy` backend. Les qtrees ne seront associés à des règles d'export restrictives de nœud qu'au moment de la publication afin d'améliorer le contrôle d'accès et la sécurité. Les qtrees existants passeront au nouveau modèle de règles d'export lorsque Trident dépubliera le volume de tous les nœuds, afin de le faire sans impacter les charges de travail actives.
- Ajout de la prise en charge de Kubernetes 1.31.

Améliorations expérimentales

- Ajout d'un aperçu technique pour la prise en charge de Fibre Channel sur le pilote ONTAP-SAN.

Correctifs

- **Kubernetes** :
  - Correction du webhook d'admission Rancher empêchant les installations de Trident Helm ("Problème n°839").
  - Clé d'affinité fixe dans les valeurs du helm chart ("Problème n°898").
  - Correction : tridentControllerPluginNodeSelector/tridentNodePluginNodeSelector ne fonctionnera pas avec la valeur "true" ("Problème n°899").
  - Suppression des instantanés éphémères créés lors du clonage ("Problème n°901").
- Ajout de la prise en charge de Windows Server 2019.
- Correction de `go mod tidy` dans le dépôt Trident ("Problème n°767").

Dépréciations

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

Changement de marque du produit

À compter de la version 24.10, Astra Trident est renommé Trident (NetApp Trident). Ce changement de nom n'affecte en rien les fonctionnalités, les plateformes prises en charge ni l'interopérabilité de Trident.

Modifications dans la version 24.06

Améliorations

- **IMPORTANT** : Le `limitVolumeSize` paramètre limite désormais la taille des qtree/LUN dans les pilotes ONTAP economy. Utilisez le nouveau `limitVolumePoolSize` paramètre pour contrôler la taille des FlexVol® dans ces pilotes. ("Problème n°341").
- Ajout de la possibilité pour l'auto-réparation iSCSI de lancer des analyses SCSI par ID LUN exact si des igroups obsolètes sont utilisés ("Problème n°883").
- Ajout de la prise en charge des opérations de clonage et de redimensionnement de volumes pour qu'elles soient autorisées même lorsque le backend est en mode suspendu.
- Ajout de la possibilité pour les paramètres de journalisation configurés par l'utilisateur pour le contrôleur Trident d'être propagés aux pods de nœud Trident.
- Ajout de la prise en charge dans Trident de l'utilisation de REST par défaut au lieu d'ONTAPI (ZAPI) pour les versions ONTAP 9.15.1 et ultérieures.
- Ajout de la prise en charge des noms de volumes personnalisés et des métadonnées sur les backends de stockage ONTAP pour les nouveaux volumes persistants.
- Amélioration du `azure-netapp-files` (ANF) pour activer automatiquement le répertoire de snapshot par défaut lorsque les options de montage NFS sont configurées pour utiliser la version NFS 4.x.
- Ajout de la prise en charge de Bottlerocket pour les volumes NFS.
- Ajout d'une prise en charge en avant-première technique pour Google Cloud NetApp Volumes.

Kubernetes

- Ajout de la prise en charge de Kubernetes 1.30.
- Ajout de la possibilité pour Trident DaemonSet de nettoyer les montures zombies et les fichiers de suivi résiduels au démarrage ("Problème n°883").
- Ajout de l'annotation PVC `trident.netapp.io/luksEncryption` pour l'importation dynamique de volumes LUKS ("Problème n°849").
- Ajout de la prise en compte de la topologie au pilote ANF.
- Ajout de la prise en charge des nœuds Windows Server 2022.

Correctifs

- Correction des échecs d'installation de Trident dus à des transactions obsolètes.
- Correction de tridentctl pour ignorer les messages d'avertissement de Kubernetes ("Problème n°892").
- Priorité du contrôleur Trident `SecurityContextConstraint` modifiée à `0` ("Problème n°887").
- Les pilotes ONTAP acceptent désormais les tailles de volume inférieures à 20 Mio ("Problème[#885").
- Correction de Trident pour empêcher la réduction des volumes FlexVol lors de l'opération de redimensionnement pour le pilote ONTAP-SAN.
- Correction de l'échec d'importation de volume ANF avec NFS v4.1.

Modifications dans la version 24.02

Améliorations

- Ajout de la prise en charge de Cloud Identity.
  - AKS avec ANF - Azure Workload Identity sera utilisé comme identité cloud.
  - EKS avec FSxN - Le rôle AWS IAM sera utilisé comme identité Cloud.
- Ajout de la prise en charge de l'installation de Trident en tant qu'add-on sur un cluster EKS depuis la console EKS.
- Ajout de la possibilité de configurer et de désactiver l'auto-réparation iSCSI ("Problème n°864").
- Ajout de la personnalité Amazon FSx aux pilotes ONTAP pour permettre l'intégration avec AWS IAM et SecretsManager, et pour permettre à Trident de supprimer les volumes FSx avec des sauvegardes ("Problème n°453").

Kubernetes

- Ajout de la prise en charge de Kubernetes 1.29.

Correctifs

- Messages d'avertissement ACP corrigés lorsque l'ACP n'est pas activé ("Problème n°866").
- Ajout d'un délai de 10 secondes avant d'effectuer une division de clone lors de la suppression d'un instantané pour les pilotes ONTAP, lorsqu'un clone est associé à l'instantané.

Dépréciations

- Suppression du framework d'attestations in-toto des manifestes d'images multiplateformes.

Changements dans la version 23.10

Correctifs

- Extension de volume fixe si une nouvelle taille demandée est inférieure à la taille totale du volume pour les pilotes de stockage ontap-nas et ontap-nas-flexgroup ("Problème n°834").
- Taille de volume fixe pour n'afficher que la taille utilisable du volume lors de l'importation pour les pilotes de stockage ontap-nas et ontap-nas-flexgroup ("Problème n° 722").
- Correction de la conversion de nom FlexVol pour ONTAP-NAS-Economy.
- Correction d'un problème d'initialisation de Trident sur un nœud Windows lorsque le nœud est redémarré.

Améliorations

Kubernetes

Ajout de la prise en charge de Kubernetes 1.28.

Trident

- Ajout de la prise en charge de l'utilisation des Azure Managed Identities (AMI) 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 backend est mis en état

suspendu par l'utilisateur ("Problème n°558").

Modifications dans 23.07.1

Kubernetes : Correction de la suppression du daemonset pour prendre en charge les mises à niveau sans interruption de service ("Problème n°740").

Modifications dans 23.07

Correctifs

Kubernetes

- Correction de la mise à niveau de Trident pour ignorer les anciens pods bloqués dans l'état de terminaison ("Problème n°740").
- Ajout d'une tolérance à la définition de "transient-trident-version-pod" ("Problème n°795").

Trident

- Requêtes ONTAPI (ZAPI) corrigées pour garantir que les numéros de série LUN sont interrogés lors de l'obtention des attributs LUN afin d'identifier et de corriger les périphériques iSCSI fantômes lors des opérations de préparation des nœuds.
- Correction de la gestion des erreurs dans le code du pilote de stockage ("Problème n°816").
- Correction du redimensionnement des quotas lors de l'utilisation des pilotes ONTAP avec use-rest=true.
- Correction de la création de clones LUN dans ontap-san-economy.
- Rétablir le champ d'informations de publication de `rawDevicePath` à `devicePath`; ajout d'une logique pour remplir et récupérer (dans certains cas) `devicePath` le champ.

Améliorations

Kubernetes

- Ajout de la prise en charge de l'importation d'instantanés pré-provisionnés.
- Déploiement minimal et permissions Linux du DaemonSet ("Problème n°817").

Trident

- Le champ d'état n'est plus signalé pour les volumes et les instantanés "online".
- Met à jour l'état du backend si le backend ONTAP est hors ligne ("Problèmes #801", "#543").
- Le numéro de série LUN est toujours récupéré et publié pendant le workflow ControllerVolumePublish.
- Ajout d'une logique supplémentaire pour vérifier le numéro de série et la taille du périphérique iSCSI multipath.
- Vérification supplémentaire des volumes iSCSI pour s'assurer que le périphérique multipath correct est désactivé.

Amélioration expérimentale

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

Documentation

De nombreuses améliorations organisationnelles et de mise en forme ont été apportées.

Dépréciations

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 minimale de Kubernetes prise en charge à 1.22.

Modifications dans la version 23.04

> Le détachement forcé des volumes ONTAP-SAN-* est uniquement pris en charge par les versions de Kubernetes dont la fonctionnalité d'arrêt non progressif des nœuds est activée. Le détachement forcé doit être activé lors de l'installation à l'aide du paramètre de l'installateur `--enable-force-detach` Trident.

Correctifs

- Correction de l'opérateur Trident pour utiliser l'localhost IPv6 pour l'installation lorsque cela est spécifié dans la spécification.
- Les autorisations du rôle de cluster de l'opérateur Trident ont été corrigées pour être synchronisées avec les autorisations du bundle ("Problème n°799").
- Correction d'un problème lié à l'attachement d'un volume de blocs bruts sur plusieurs nœuds en mode RWX.
- Correction de la prise en charge du clonage FlexGroup et de l'importation de volumes SMB.
- Correction d'un problème où le contrôleur Trident ne pouvait pas s'éteindre immédiatement ("Problème n°811").
- Ajout d'un correctif pour lister tous les noms d'igroup associés à un LUN spécifié provisionné avec les pilotes ontap-san-*.
- Ajout d'un correctif pour permettre aux processus externes de s'exécuter jusqu'à leur terme.
- Correction d'une erreur de compilation pour l'architecture s390 ("Problème n° 537").
- Correction du niveau de journalisation incorrect lors des opérations de montage de volume ("Problème n°781").
- Correction d'une erreur potentielle d'assertion de type ("Problème n°802").

Améliorations

- Kubernetes :
  - Ajout de la prise en charge de Kubernetes 1.27.
  - Ajout de la prise en charge de l'importation des 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 non progressif des nœuds.

- Tous les volumes ONTAP-SAN-* utiliseront désormais des igroups par nœud. Les LUN ne seront mappés aux igroups que lorsqu'ils seront activement publiés sur ces nœuds afin d'améliorer notre posture de sécurité. Les volumes existants seront basculés de manière opportuniste vers le nouveau schéma d'igroups lorsque Trident déterminera qu'il est sûr de le faire sans impacter les charges de travail actives ("Problème n°758").
- Amélioration de la sécurité de Trident grâce au nettoyage des igroups gérés par Trident inutilisés sur les backends 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 ("Problème n°732").
- Procédure d'arrêt Trident améliorée en désactivant d'abord les serveurs API ("Problème n°811").
- Ajout de la prise en charge de la compilation multiplateforme pour les hôtes Windows et arm64 au Makefile ; voir BUILD.md.

Dépréciations

Kubernetes: Les igroups à portée backend ne seront plus créés lors de la configuration des pilotes ontap-san et ontap-san-economy ("Problème n°758").

Modifications dans 23.01.1

Correctifs

- Correction de l'opérateur Trident pour utiliser l'localhost IPv6 pour l'installation lorsque cela est spécifié dans la spécification.
- Correction des autorisations du rôle de cluster de l'opérateur Trident pour qu'elles soient synchronisées avec les autorisations du bundle "Problème n°799".
- Ajout d'un correctif pour permettre aux processus externes de s'exécuter jusqu'à leur terme.
- Correction d'un problème lié à l'attachement d'un volume de blocs bruts sur plusieurs nœuds en mode RWX.
- Correction de la prise en charge du clonage FlexGroup et de l'importation de volumes SMB.

Modifications dans 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 Pod Security Policy afin de corriger les installations de Trident via Helm ("Problèmes #783, #794").

Améliorations

Kubernetes

- Ajout de la prise en charge de Kubernetes 1.26.

- Amélioration de l'utilisation globale des ressources Trident RBAC ("Problème n°757").
- Ajout de l'automatisation 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 niveau de service CVS.
- Ajout de la prise en charge de l'utilisation optionnelle de flexgroupAggregateList lors de la création de FlexGroups avec le pilote de stockage ontap-nas-flexgroup.
- Amélioration des performances du pilote de stockage ontap-nas-economy lors de la gestion de plusieurs volumes FlexVol
- Activation des mises à jour dataLIF pour tous les pilotes de stockage NAS ONTAP.
- Mise à jour de la convention d'appellation du déploiement Trident et DaemonSet pour refléter le système d'exploitation du nœud hôte.

Dépréciations

- Kubernetes: Mise à jour de la version minimale de Kubernetes prise en charge à 1.21.
- Les DataLIF ne doivent plus être spécifiés lors de la configuration `ontap-san` ou `ontap-san-economy` des pilotes.

Changements dans 22.10

Vous devez lire les informations critiques suivantes avant de procéder à la mise à niveau vers Trident 22.10.

Informations essentielles sur Trident 22.10

- Kubernetes 1.25 est désormais pris en charge dans Trident. Vous devez mettre à niveau Trident vers la version 22.10 avant de procéder à la mise à niveau vers Kubernetes 1.25.
- Trident impose désormais strictement l'utilisation de la configuration multipath dans les environnements SAN, avec une valeur recommandée de `find_multipaths: no` dans le fichier multipath.conf.

  L'utilisation d'une configuration sans multipath ou l'utilisation de `find_multipaths: yes` ou `find_multipaths: smart` dans le 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 corrigé pour le backend ONTAP créé à l'aide du champ `credentials` qui ne parvenait pas à se mettre en ligne lors de la mise à niveau 22.07.0 ("Problème n°759").

- **Docker:** Correction d'un problème empêchant le démarrage du plugin de volume Docker dans certains environnements ("Problème n°548" et "Problème n°760").
- Correction d'un problème SLM spécifique aux backends SAN ONTAP afin de garantir que seul un sous-ensemble des dataLIF appartenant aux nœuds de reporting soit publié.
- Correction d'un problème de performances où des analyses inutiles des LUN iSCSI se produisaient lors de l'attachement d'un volume.
- Suppression des tentatives granulaires de nouvelle tentative dans le workflow iSCSI de Trident pour détecter rapidement les échecs et réduire les intervalles de nouvelle tentative externes.
- Correction d'un problème où une erreur était renvoyée lors du vidage d'un périphérique iSCSI lorsque le périphérique multipath correspondant avait déjà été vidé.

Améliorations

- Kubernetes :
  - Ajout de la prise en charge de Kubernetes 1.25. Vous devez mettre à niveau Trident vers la version 22.10 avant de procéder à la mise à niveau vers Kubernetes 1.25.
  - Ajout d'un ServiceAccount, d'un ClusterRole et d'un ClusterRoleBinding distincts pour le déploiement Trident et le DaemonSet afin de permettre de futures améliorations des autorisations.
  - Ajout de la prise en charge de "partage de volumes entre espaces de noms".
- Tous les pilotes de stockage `ontap-*`Trident fonctionnent désormais avec l'API REST ONTAP.
- Ajout d'un nouvel opérateur yaml (`bundle_post_1_25.yaml`) sans `PodSecurityPolicy` pour prendre en charge Kubernetes 1.25.
- Ajouté "prise en charge des volumes chiffrés LUKS" pour `ontap-san` et `ontap-san-economy` les pilotes de stockage.
- Ajout de la prise en charge des nœuds Windows Server 2019.
- Ajouté "prise en charge des volumes SMB sur les nœuds Windows" through the `azure-netapp-files` pilote de stockage.
- La détection automatique du basculement MetroCluster pour les pilotes ONTAP est désormais disponible de manière générale.

Dépréciations

- **Kubernetes :** Mise à jour de la version minimale de Kubernetes prise en charge à 1.20.
- Pilote Astra Data Store (ADS) supprimé.
- Suppression de la prise en charge de `yes` et `smart` options pour `find_multipaths` lors de la configuration du multipathing des nœuds de travail pour iSCSI.

Modifications dans la version 22.07

Correctifs

Kubernetes

- Correction d'un problème de gestion des valeurs booléennes et numériques pour le sélecteur de nœuds lors de la configuration de Trident avec Helm ou le Trident Operator. ("Problème GitHub #700")
- Correction d'un problème de gestion des erreurs provenant d'un chemin non-CHAP, de sorte que kubelet

réessaiera en cas d'échec. "Problème GitHub #736")

Améliorations

- Transition de k8s.gcr.io à registry.k8s.io comme registre par défaut pour les images CSI
- Les volumes ONTAP-SAN utiliseront désormais des igroups par nœud et n'associeront les LUN à des igroups que lorsqu'ils sont activement publiés sur ces nœuds afin d'améliorer notre posture de sécurité. Les volumes existants seront migrés de manière opportuniste vers le nouveau système d'igroups lorsque Trident déterminera qu'il est possible de le faire sans impacter les charges de travail actives.
- Inclus un ResourceQuota avec les installations Trident pour garantir que le DaemonSet Trident est planifié lorsque la consommation de PriorityClass est limitée par défaut.
- Ajout de la prise en charge des fonctionnalités réseau au pilote Azure NetApp Files. ("Problème GitHub n°717")
- Ajout de la prévisualisation technique de la détection automatique du basculement MetroCluster aux pilotes ONTAP. ("Problème GitHub #228")

Dépréciations

- **Kubernetes :** Mise à jour de la version minimale de Kubernetes prise en charge à 1.19.
- La configuration du backend n'autorise plus plusieurs types d'authentification dans une seule configuration.

Suppressions

- Le pilote AWS CVS (déprécié depuis 22.04) a été supprimé.
- Kubernetes
  - Suppression de la capacité SYS_ADMIN inutile des pods de nœud.
  - Réduit nodeprep à de simples informations sur l'hôte et à une découverte active des services pour faire une confirmation au mieux que les services NFS/iSCSI sont disponibles sur les nœuds de travail.

Documentation

Une nouvelle section "Normes de sécurité des pods" (PSS) a été ajoutée, détaillant les permissions activées par Trident lors de l'installation.

Modifications dans 22.04

NetApp améliore et perfectionne continuellement ses produits et services. Voici quelques-unes des dernières fonctionnalités de Trident. Pour les versions précédentes, consultez "Versions antérieures de la documentation".

> Si vous effectuez une mise à niveau depuis une version précédente de Trident et que vous utilisez Azure NetApp Files, le `location` paramètre de configuration est désormais un champ unique et obligatoire.

Correctifs

- Amélioration de l'analyse des noms d'initiateurs iSCSI. ("Problème GitHub #681")
- Correction d'un problème où les paramètres de classe de stockage CSI n'étaient pas autorisés. ("Problème GitHub n°598")

- Correction d'une déclaration de clé dupliquée dans la CRD Trident. ("Problème GitHub #671")
- Correction des journaux d'instantanés CSI inexacts. ("Problème GitHub #629"))
- Correction d'un problème lié à la dépublication des volumes sur les nœuds supprimés. ("Problème GitHub #691")
- Ajout de la gestion des incohérences du système de fichiers sur les périphériques de stockage par blocs. ("Problème GitHub #656")
- Correction d'un problème d'extraction des images auto-support lors de la définition du `imageRegistry` flag pendant l'installation. ("Problème GitHub #715")
- Correction d'un problème où le pilote Azure NetApp Files ne parvenait pas à cloner un volume comportant plusieurs règles d'exportation.

Améliorations

- Les connexions entrantes aux points de terminaison sécurisés de Trident nécessitent désormais au minimum TLS 1.3. ("Problème GitHub #698")
- Trident ajoute désormais des en-têtes HSTS aux réponses de ses points de terminaison sécurisés.
- Trident tente désormais d'activer automatiquement la fonctionnalité d'autorisations Unix Azure NetApp Files.
- **Kubernetes** : Le daemonset Trident s'exécute désormais avec la classe de priorité system-node-critical. ("Problème GitHub #694")

Suppressions

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

Modifications dans 22.01.1

Correctifs

- Correction d'un problème lié à la dépublication des volumes sur les nœuds supprimés. ("Problème GitHub #691")
- Correction d'un problème de panique lors de l'accès aux champs nuls pour l'espace agrégé dans les réponses de l'API ONTAP.

Modifications dans la version 22.01.0

Correctifs

- **Kubernetes:** Augmenter le délai de nouvelle tentative d'enregistrement des nœuds pour les grands clusters.
- Correction d'un problème où le pilote azure-netapp-files pouvait être perturbé par plusieurs ressources portant le même nom.
- Les DataLIFs ONTAP SAN IPv6 fonctionnent désormais si elles sont spécifiées avec des crochets.
- Correction d'un problème où la tentative d'importation d'un volume déjà importé renvoie EOF, laissant le PVC en état d'attente. ("Problème GitHub #489")
- Correction d'un problème où les performances de Trident ralentissent lorsque plus de 32 instantanés sont créés sur un volume SolidFire.

- SHA-1 remplacé par SHA-256 lors de la création du certificat SSL.
- Correction du pilote Azure NetApp Files pour autoriser les noms de ressources en double et limiter les opérations à un seul emplacement.
- Correction du pilote Azure NetApp Files pour autoriser les noms de ressources en double et limiter les opérations à un seul emplacement.

Améliorations

- Améliorations de Kubernetes :
  - Ajout de la prise en charge de Kubernetes 1.23.
  - Ajouter des options de planification pour les pods Trident lors de leur installation via Trident Operator ou Helm. ("Problème GitHub #651")
- Autoriser les volumes interrégionaux dans le pilote GCP. ("Problème GitHub #633")
- Ajout de la prise en charge de l'option « unixPermissions » pour les volumes Azure NetApp Files. ("Problème GitHub #666")

Dépréciations

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

Modifications dans 21.10.1

La version v21.10.0 présente un problème qui peut mettre le contrôleur Trident dans un état CrashLoopBackOff lorsqu'un nœud est supprimé puis réintégré au cluster Kubernetes. Ce problème est corrigé dans la version v21.10.1 (GitHub issue 669).

Correctifs

- Correction d'une condition de concurrence potentielle lors de l'importation d'un volume sur un backend GCP CVS, entraînant un échec de l'importation.
- Correction d'un problème qui peut mettre le contrôleur Trident dans un état CrashLoopBackOff lorsqu'un nœud est supprimé puis ajouté à nouveau au cluster Kubernetes (GitHub issue 669).
- Correction d'un problème où les SVM n'étaient plus détectés si aucun nom de SVM n'était spécifié (GitHub issue 612).

Modifications dans la version 21.10.0

Correctifs

- Correction d'un problème où les clones de volumes XFS ne pouvaient pas être montés sur le même nœud que le volume source (GitHub issue 514).
- Correction d'un problème où Trident enregistrait une erreur fatale lors de l'arrêt (GitHub issue 597).
- Correctifs liés à Kubernetes :
  - Renvoyer l'espace utilisé d'un volume comme minimum restoreSize lors de la création d'instantanés avec `ontap-nas` et `ontap-nas-flexgroup` drivers (GitHub issue 645).
  - Correction d'un problème où `Failed to expand filesystem` une erreur était consignée après le redimensionnement du volume (GitHub issue 560).

- Correction d'un problème où un pod pouvait rester bloqué dans l'état `Terminating` (GitHub issue 572).
- Correction du cas où un `ontap-san-economy` FlexVol pouvait être plein de LUN instantanés (GitHub issue 533).
- Correction d'un problème d'installation YAML personnalisé avec une image différente (GitHub issue 613).
- Calcul corrigé de la taille des instantanés (GitHub issue 611).
- Correction d'un problème où tous les installateurs Trident pouvaient identifier Kubernetes simple comme OpenShift (GitHub issue 639).
- Correction de l'opérateur Trident pour arrêter la réconciliation si le serveur API Kubernetes est inaccessible (GitHub issue 599).

Améliorations

- Ajout de la prise en charge de l' `unixPermissions`option pour les volumes GCP-CVS Performance.
- Ajout de la prise en charge des volumes CVS optimisés pour l'échelle dans GCP, dans la plage de 600 GiB à 1 TiB.
- Améliorations liées à Kubernetes :
  - Ajout de la prise en charge de Kubernetes 1.22.
  - Activation de l'opérateur Trident et du chart Helm pour fonctionner avec Kubernetes 1.22 (GitHub issue 628).
  - Ajout de l'image de l'opérateur à la commande `tridentctl` images (GitHub issue 570).

Améliorations expérimentales

- Ajout de la prise en charge de la réplication de volumes dans le `ontap-san` driver.
- Ajout de la prise en charge REST **en avant-première technique** pour les `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 à 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 `--set excludePodSecurityPolicy=true` à la commande `helm upgrade` avant de pouvoir mettre à niveau le cluster.
- Trident impose désormais une valeur vide `fsType` (`fsType=""` pour les volumes qui n'ont pas le `fsType` spécifié dans leur StorageClass. Lorsque vous travaillez avec Kubernetes 1.17 ou version ultérieure, Trident prend en charge la fourniture d'une valeur vide `fsType` pour les volumes NFS. Pour les volumes iSCSI, vous devez définir le `fsType` sur votre StorageClass lors de l'application d'une `fsGroup` à l'aide d'un contexte de sécurité.
- Lors de l'utilisation d'un backend sur plusieurs instances Trident, chaque fichier de configuration backend doit avoir une valeur `storagePrefix` différente pour les backends ONTAP ou utiliser une valeur `TenantName` différente pour les backends SolidFire. Trident ne peut pas détecter les volumes créés par d'autres instances de Trident. Tenter de créer un volume existant sur les backends ONTAP ou SolidFire

réussit, car Trident considère la création de volume comme une opération idempotente. Si `storagePrefix` ou `TenantName` ne diffèrent pas, il peut y avoir des collisions de noms pour les volumes créés sur le même backend.

- Lors de l'installation de Trident (en utilisant `tridentctl` ou l'Operator Trident) et en utilisant `tridentctl` pour gérer Trident, vous devez vous assurer que la variable d'environnement `KUBECONFIG` est définie. Ceci est nécessaire pour indiquer le cluster Kubernetes auquel `tridentctl` doit se connecter. Lorsque vous travaillez avec plusieurs environnements Kubernetes, vous devez vous assurer que le fichier `KUBECONFIG` est correctement référencé.
- Pour effectuer la récupération d'espace en ligne pour les volumes persistants iSCSI, le système d'exploitation sous-jacent du nœud de travail peut exiger que des options de montage soient transmises au volume. Cela est vrai pour les instances RHEL/Red Hat Enterprise Linux CoreOS (RHCOS), qui nécessitent le `discard` "option de montage"; assurez-vous que l'option discard mountOption est incluse dans votre[`StorageClass`^] pour prendre en charge la suppression de blocs en ligne.
- Si vous avez plus d'une instance de Trident par cluster Kubernetes, Trident ne peut pas communiquer avec les autres instances et ne peut pas découvrir les autres volumes qu'elles ont créés, ce qui entraîne un comportement inattendu et incorrect si plus d'une instance s'exécute 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 pendant que Trident est hors ligne, Trident ne supprime pas les classes de stockage correspondantes de sa base de données lorsqu'il revient en ligne. Vous devez supprimer ces classes de stockage à l'aide de `tridentctl` ou de l'API REST.
- Si un utilisateur supprime un PV provisionné par Trident avant de supprimer le PVC correspondant, Trident ne supprime pas automatiquement le volume sous-jacent. Vous devez supprimer le volume via `tridentctl` ou l'API REST.
- ONTAP ne peut pas provisionner simultanément plus d'un FlexGroup à la fois, sauf si l'ensemble des agrégats est unique à chaque demande de provisionnement.
- Lors de l'utilisation de 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 backend SAN ONTAP. Trident découvre toutes les LIF iSCSI disponibles et les utilise pour établir la session multipath.

- Si vous utilisez le `solidfire-san` driver 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.

Pour plus d'informations

- "Trident GitHub"
- "Blogs de 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"

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

- chiffrement 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 SAN ONTAP".

Problèmes connus

Les problèmes connus recensent les dysfonctionnements susceptibles de vous empêcher d'utiliser correctement cette version du produit.

Les problèmes connus suivants affectent 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 avec Restic, l'opération de restauration peut échouer. Comme solution de contournement, sauvegardez les données en utilisant Kopia comme outil de déplacement de données (Kopia est l'outil de déplacement de données par défaut pour les sauvegardes). Consultez "Protégez les applications à l'aide de Trident Protect" pour obtenir des instructions.

Commencer

En savoir plus sur Trident

En savoir plus sur Trident

Trident est un projet open source entièrement pris en charge et maintenu par NetApp. Il a été conçu pour vous aider à répondre aux exigences de persistance de votre application conteneurisée en utilisant des interfaces standard du secteur, telles que l'interface de stockage de conteneurs (CSI).

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 l'interface CSI (Container Storage Interface) qui s'intègre nativement à "Kubernetes". 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. Consultez "Architecture Trident" pour plus de détails.

Trident propose également une intégration directe avec l'écosystème Docker pour les plateformes de stockage NetApp. Le NetApp Docker Volume Plugin (nDVP) prend en charge le provisionnement et la gestion des ressources de stockage depuis la plateforme de stockage vers les hôtes Docker. Consultez "Déployer Trident pour Docker" pour plus de détails.

Si c'est la première fois que vous utilisez Kubernetes, vous devriez vous familiariser 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 d'un cluster Kubernetes, offrant des capacités avancées de gestion des données qui améliorent la fonctionnalité, la capacité, les performances et la disponibilité du déploiement Kubernetes.

Amazon FSx for NetApp ONTAP

"Amazon FSx for NetApp ONTAP" est un service AWS entièrement géré qui vous permet de lancer et d'exécuter des systèmes de fichiers alimentés par le système d'exploitation de stockage NetApp ONTAP.

Azure NetApp Files

"Azure NetApp Files" est un service de partage de fichiers Azure de niveau entreprise, optimisé par NetApp. Vous pouvez exécuter vos charges de travail les plus exigeantes basées sur des fichiers dans Azure nativement, avec les performances et la gestion des données avancée que vous attendez de NetApp.

Cloud Volumes ONTAP

"Cloud Volumes ONTAP" est une appliance de stockage uniquement 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 sur Google Cloud qui fournit un stockage de fichiers haute performance et de niveau entreprise.

logiciel Element

"Élément" permet à l'administrateur de stockage de consolider les charges de travail en garantissant les performances et en permettant un encombrement de stockage simplifié et rationalisé.

NetApp HCI

"NetApp HCI" simplifie la gestion et l'échelle du centre de données 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 périphériques de stockage pour des applications conteneurisées directement sur la plateforme de stockage NetApp HCI sous-jacente.

NetApp ONTAP

"NetApp ONTAP" est le système d'exploitation de stockage multiprotocole et unifié NetApp qui offre des capacités avancées de gestion des données pour toute application.

Les systèmes ONTAP proposent des configurations 100 % flash, hybrides ou 100 % HDD et offrent de nombreux modèles de déploiement : FAS sur site, AFA et ASA clusters, ONTAP Select et Cloud Volumes ONTAP. Trident prend en charge ces modèles de déploiement ONTAP.

Architecture Trident

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 de nœud doit être exécuté sur tout hôte sur lequel vous souhaitez potentiellement monter un volume Trident.

Comprendre les pods de contrôleur et les pods de nœud

Trident se déploie en tant qu'Pod du contrôleur Trident unique et un ou plusieurs Modules de nœuds Trident sur le cluster Kubernetes et utilise les conteneurs *CSI Sidecar Containers* standard de Kubernetes pour simplifier le déploiement des plugins CSI. "Conteneurs sidecar CSI Kubernetes" sont maintenus par la communauté Kubernetes Storage.

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

- Le plugin de contrôleur gère le provisionnement et la gestion des volumes, comme les instantanés et le redimensionnement.
- Le plugin de nœud gère la connexion du stockage au nœud.

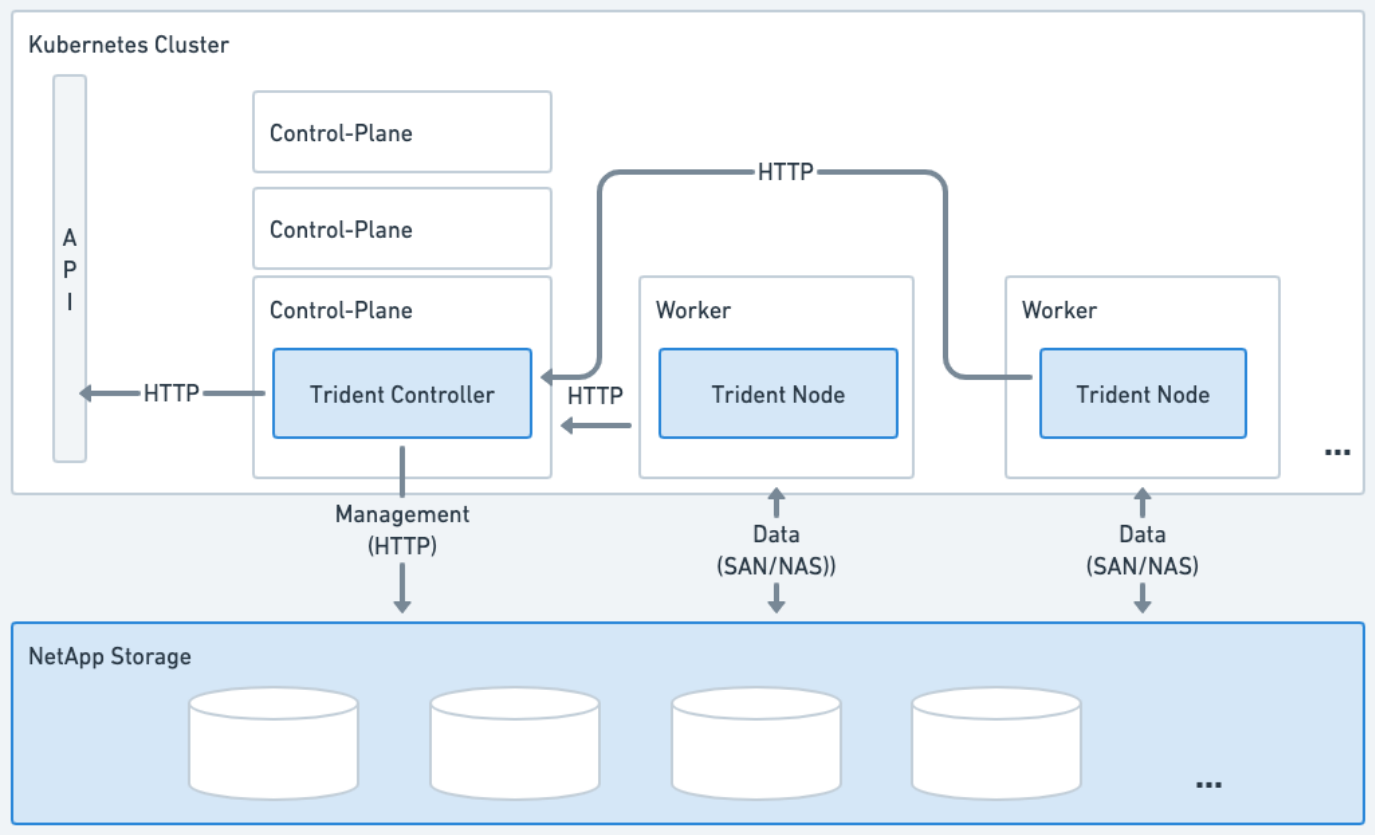


Figure 1. Trident déployé sur le cluster Kubernetes

Pod du contrôleur Trident

Le Trident Controller Pod est un pod unique exécutant le plugin CSI Controller.

- Responsable de la mise en service et de la gestion des volumes dans le stockage NetApp
- Géré par un déploiement Kubernetes
- Peut s'exécuter sur le plan de contrôle ou sur les nœuds de travail, selon les paramètres d'installation.

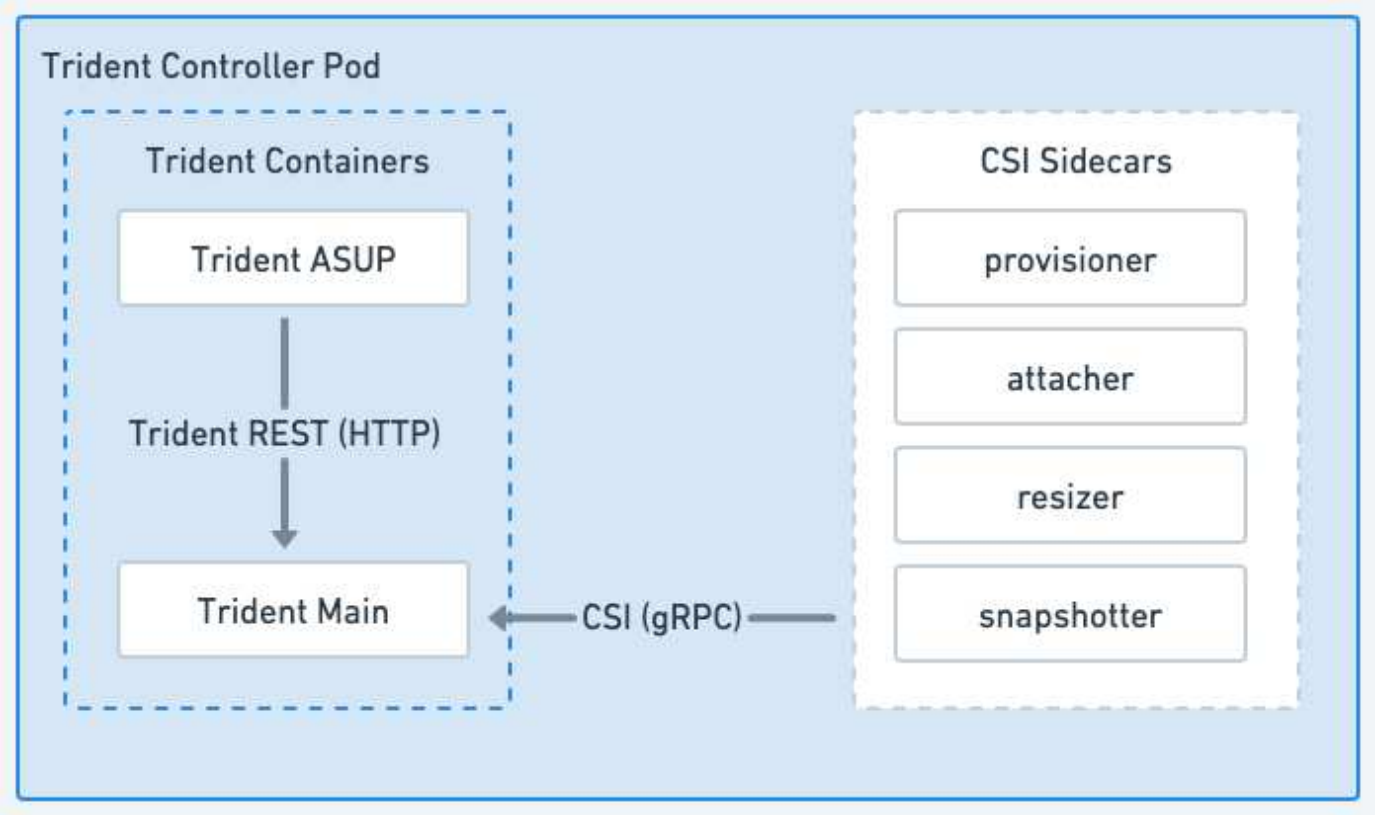


Figure 2. Schéma du module de commande Trident

Modules de nœuds Trident

Les pods Trident Node sont des pods privilégiés exécutant le plugin CSI Node.

- Responsable du montage et du démontage du stockage pour les Pods exécutés sur l'hôte
- Géré par un Kubernetes DaemonSet
- Doit être exécuté sur n'importe quel nœud qui montera le stockage NetApp

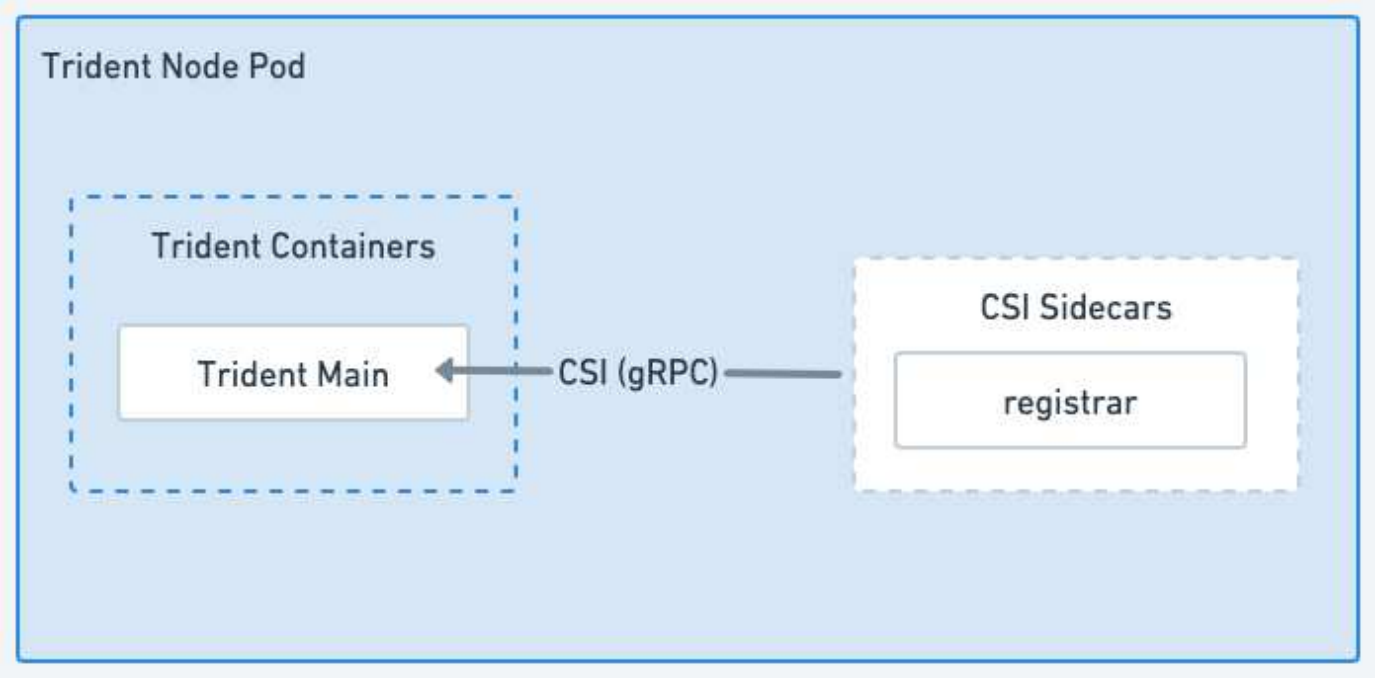


Figure 3. Diagramme du nœud Trident

Architectures de cluster Kubernetes prises en charge

Trident est compatible avec les architectures Kubernetes suivantes :

Architectures de clusters Kubernetes	Pris en charge	Installation par défaut
Maître unique, calcul	Oui	Oui
Plusieurs maîtres, compute	Oui	Oui
Maître, `etcd` calcul	Oui	Oui
Maître, 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 backend appropriés et se produit comme une préparation nécessaire avant le provisionnement. La seconde phase inclut la création du volume elle-même et nécessite de choisir 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 backend à une classe de stockage dépend à la fois des attributs requis par la classe de stockage et de ses `storagePools`, `additionalStoragePools`, et `excludeStoragePools` listes. Lorsque vous créez une classe de stockage, Trident compare les attributs et les pools proposés par

chacun de ses backends à ceux demandés par la classe de stockage. Si les attributs et le nom d'un pool de stockage correspondent à tous les attributs et noms de pools demandés, Trident ajoute ce pool de stockage à l'ensemble des pools de stockage adaptés à cette classe de stockage. De plus, Trident ajoute à cet ensemble tous les pools de stockage listés dans la `additionalStoragePools` liste, même si leurs attributs ne remplissent pas tous ou aucun des attributs demandés par la classe de stockage. Vous devez utiliser la `excludeStoragePools` liste pour exclure et retirer des pools de stockage de l'utilisation pour une classe de stockage. Trident effectue un processus similaire chaque fois que vous ajoutez un nouveau backend, en vérifiant si ses pools de stockage satisfont ceux des classes de stockage existantes et en retirant ceux qui ont été marqués comme exclus.

Création de 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 récupère d'abord l'ensemble des pools de stockage pour la classe de stockage de ce volume et, 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 backend NetApp HCI/SolidFire ne peut pas fournir un volume basé sur des fichiers, tandis qu'un backend ONTAP NAS ne peut pas fournir un volume basé sur des blocs). Trident randomise l'ordre de cet ensemble résultant, afin de faciliter une répartition homogène des volumes, puis le parcourt en tentant de provisionner le volume sur chaque pool de stockage à son tour. S'il réussit sur l'un d'eux, il retourne un succès, en enregistrant tous les échecs rencontrés au cours du processus. Trident signale un échec **uniquement si** le provisionnement échoue sur **tous** les pools de stockage disponibles pour la classe de stockage et le protocole demandés.

Instantanés de volume

Découvrez comment Trident gère la création d'instantanés de volume pour ses pilotes.

Découvrez la création d'instantanés de volume

- Pour les `ontap-nas`, `ontap-san` et `azure-netapp-files` pilotes, chaque volume persistant (PV) est associé à un volume FlexVol. Par conséquent, les instantanés de volume sont créés sous forme d'instantanés NetApp. La technologie d'instantanés NetApp offre plus de stabilité, de scalabilité, de capacité de récupération et de performance que les technologies d'instantanés concurrentes. Ces copies instantanées sont extrêmement efficaces, tant en temps de création qu'en espace de stockage.
- Pour le `ontap-nas-flexgroup` driver, chaque volume persistant (PV) correspond à un FlexGroup. Par conséquent, les instantanés de volume sont créés en tant qu'instantanés NetApp FlexGroup. La technologie d'instantanés NetApp offre plus de stabilité, de scalabilité, de capacité de récupération et de performance que les technologies d'instantanés concurrentes. Ces copies instantanées sont extrêmement efficaces, tant en temps de création qu'en espace de stockage.
- Pour le `ontap-san-economy` pilote, les PV sont associés à des LUN créées sur des volumes FlexVol partagés. Les VolumeSnapshots des PV sont obtenus en effectuant des FlexClones de la LUN associée. La technologie ONTAP FlexClone permet de créer des copies, même des plus grands ensembles de données, presque instantanément. Les copies partagent des blocs de données avec leurs parents, ne consommant aucun espace de stockage à l'exception de celui requis pour les métadonnées.
- Pour le `solidfire-san` pilote, chaque PV correspond à un LUN créé sur le logiciel NetApp Element/NetApp HCI cluster. VolumeSnapshots sont représentés par des instantanés Element du LUN sous-jacent. Ces instantanés sont des copies à un instant précis et n'occupent qu'une faible quantité de ressources système et d'espace.
- Lors de l'utilisation des `ontap-nas` et `ontap-san` pilotes, les snapshots ONTAP sont des copies instantanées du FlexVol et consomment de l'espace sur le FlexVol lui-même. Cela peut entraîner une réduction de l'espace inscriptible dans le volume au fil du temps à mesure que des snapshots sont créés ou planifiés. Une façon simple de résoudre ce problème consiste à agrandir le volume en le

redimensionnant via Kubernetes. Une autre option consiste à supprimer les snapshots qui ne sont plus nécessaires. Lorsqu'un VolumeSnapshot créé via Kubernetes est supprimé, Trident supprimera le snapshot ONTAP associé. Les snapshots ONTAP qui n'ont pas été créés via Kubernetes peuvent également être supprimés.

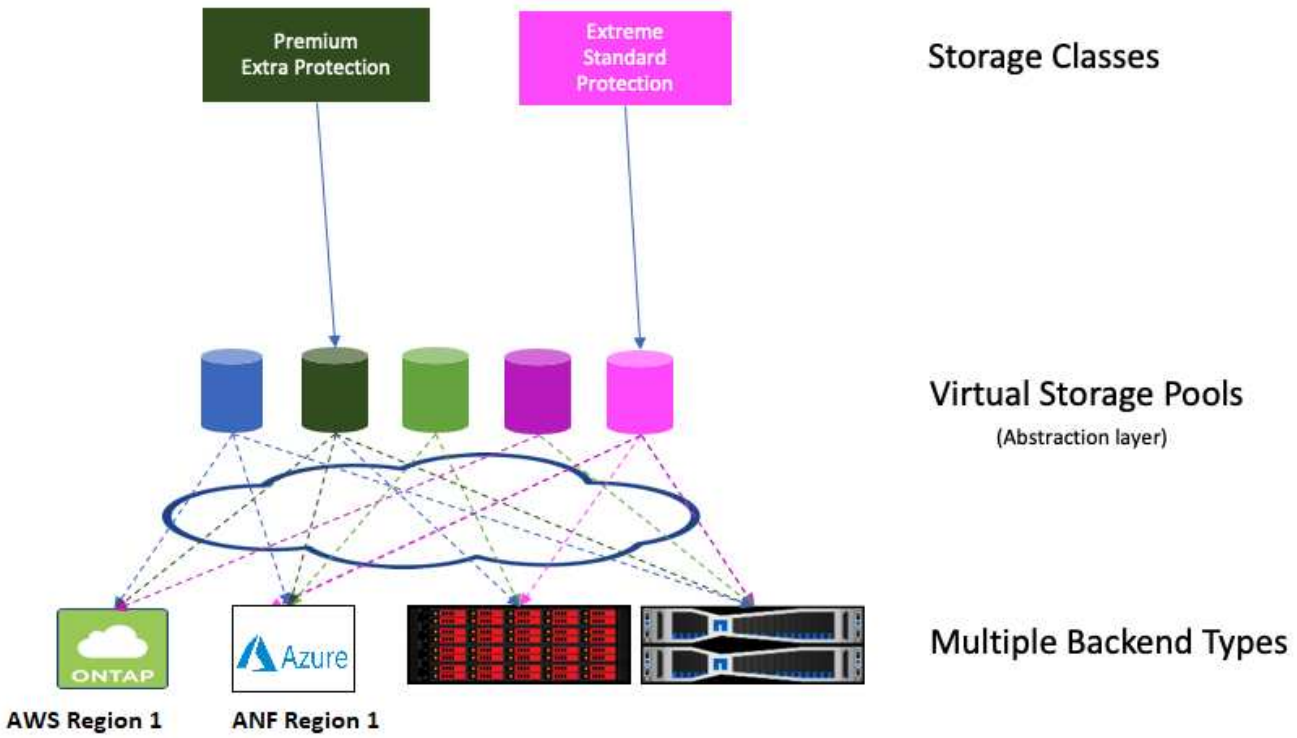
Avec Trident, vous pouvez utiliser VolumeSnapshots pour créer de nouveaux PV à partir de celles-ci. La création de PV à partir de ces snapshots est effectuée en utilisant la technologie FlexClone pour les backends ONTAP pris en charge. Lors de la création d'un PV à partir d'un snapshot, le volume de base est un FlexClone du volume parent du snapshot. Le `solidfire-san` driver utilise les clones de volumes logiciels Element pour créer des PV à partir de snapshots. Ici, il crée un clone à partir du snapshot Element.

Pools virtuels

Les pools virtuels offrent une couche d'abstraction entre Trident storage backends et Kubernetes `StorageClasses`. Ils permettent à un administrateur de définir des aspects, tels que l'emplacement, les performances et la protection pour chaque backend de manière commune et indépendante du backend, sans avoir à `StorageClass` spécifier quel backend physique, pool de backend ou type de backend utiliser pour répondre aux critères souhaités.

En savoir plus sur les pools virtuels

L'administrateur de stockage peut définir des pools virtuels sur n'importe quel backend Trident dans un fichier de définition JSON ou YAML.



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

(remplaçant tout aspect global au backend).

- Lors de la définition des pools virtuels, ne tentez pas de réorganiser l'ordre des pools virtuels existants dans une définition de backend.
- Nous déconseillons de modifier les attributs d'un pool virtuel existant. Vous devez définir un nouveau pool virtuel pour effectuer des modifications.

La plupart des aspects sont spécifiés en termes propres au backend. Il est essentiel de noter que les valeurs des aspects ne sont pas exposées en dehors du pilote du backend et ne sont pas disponibles pour la mise en correspondance dans `StorageClasses`. À la place, l'administrateur définit une ou plusieurs étiquettes pour chaque pool virtuel. Chaque étiquette est une paire clé:valeur, et les étiquettes peuvent être communes à différents backends uniques. Comme les aspects, les étiquettes peuvent être spécifiées par pool ou globalement pour le backend. Contrairement aux aspects, qui ont des noms et des valeurs prédéfinis, l'administrateur a toute latitude pour définir les clés et les valeurs des étiquettes selon les besoins. Pour plus de simplicité, les administrateurs de stockage peuvent définir des étiquettes par pool virtuel et regrouper les volumes par étiquette.

Les étiquettes de pool virtuel peuvent être définies à l'aide de ces caractères :

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

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

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

Groupes d'accès au volume

Apprenez-en davantage sur la façon dont Trident utilise "groupes d'accès au 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 provisionneur CSI amélioré.

En savoir plus sur les groupes d'accès au volume

Trident peut utiliser des groupes d'accès aux volumes 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.

Bien que Trident associe les nouveaux volumes aux groupes d'accès configurés, il ne crée ni ne gère les groupes d'accès eux-mêmes. Les groupes d'accès doivent exister avant que le backend de stockage ne soit ajouté à Trident, et ils doivent contenir les IQN iSCSI de chaque nœud du cluster Kubernetes susceptible de monter les volumes provisionnés par ce backend. Dans la plupart des installations, cela inclut chaque nœud de travail du 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 les quatre groupes d'accès configurés au maximum, tout nœud d'un cluster pouvant atteindre 256 nœuds pourra accéder à n'importe quel volume. Pour connaître les dernières limites concernant les groupes d'accès aux volumes, consultez "ici".

Si vous modifiez la configuration d'une configuration utilisant le groupe d'accès par défaut `trident` à une configuration utilisant également d'autres groupes, incluez l'ID du groupe d'accès `trident` 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 "Exigences de Trident".

> Pour Docker, veuillez vous référer à "Trident for Docker".

1 Préparer le nœud de travail

Tous les nœuds de travail du cluster Kubernetes doivent pouvoir monter les volumes que vous avez provisionnés pour vos pods.

"Préparez le nœud de travail"

2 Installer Trident

Trident propose plusieurs méthodes et modes d'installation optimisés pour une variété d'environnements et d'organisations.

"Installer Trident"

3 Créer un backend

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

"Configurer un backend" for your système de stockage

4 Créez une StorageClass Kubernetes

L'objet Kubernetes StorageClass 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 provisionneur Trident.

"Créer une classe de stockage"

5 Provisionnez un volume

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

Créez un PersistentVolume (PV) et un PersistentVolumeClaim (PVC) qui utilise la StorageClass Kubernetes configurée pour demander l'accès au PV. Vous pouvez ensuite monter le PV sur un pod.

"Provisionner un volume"

Et ensuite ?

Vous pouvez désormais ajouter des backends supplémentaires, gérer les classes de stockage, gérer les backends et effectuer des opérations sur les volumes.

Exigences

Avant d'installer Trident, vous devez examiner ces exigences générales du système. Certains backends peuvent avoir des exigences supplémentaires.

Informations essentielles sur Trident

Vous devez lire les informations essentielles suivantes concernant Trident.

Informations essentielles sur Trident

- Kubernetes 1.36 est désormais pris en charge dans Trident. Mettez à niveau Trident avant de mettre à niveau Kubernetes.
- Trident impose strictement l'utilisation de la configuration multipath dans les environnements SAN, avec une valeur recommandée de `find_multipaths: no` dans le fichier multipath.conf.

  L'utilisation d'une configuration sans multipath ou l'utilisation de `find_multipaths: yes` ou `find_multipaths: smart` dans le fichier multipath.conf entraînera des échecs de montage. Trident a recommandé l'utilisation de `find_multipaths: no` depuis la version 21.07.

Interfaces prises en charge (orchestrators)

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

- Anthos On-Prem (VMware) et Anthos on bare metal 1.16
- Kubernetes 1.27 - 1.36
- OpenShift 4.12, 4.14 - 4.21 (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 de OpenShift en accord avec le "Cycle de vie des versions Red Hat Extended Update Support (EUS)", même si elles reposent 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 sans risque 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, y compris Google Kubernetes Engine (GKE), Amazon Elastic Kubernetes Services (EKS), Azure Kubernetes Service (AKS), Mirantis Kubernetes Engine (MKE) et VMWare Tanzu Portfolio.

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 à la version 1.26 ou ultérieure sur lequel Trident est installé, reportez-vous à "Mettre à niveau une installation Helm".

Backends pris en charge (stockage)

Pour utiliser Trident, vous avez besoin d'un ou plusieurs des backends pris en charge suivants :

- Amazon FSx for NetApp ONTAP
- Azure NetApp Files
- Cloud Volumes ONTAP
- Google Cloud NetApp Volumes
- NetApp All SAN Array (ASA)
- FAS, AFF ou ASA r2 sur site (iSCSI, NVMe/TCP et FC) exécutant des versions d'ONTAP bénéficiant d'une prise en charge complète ou limitée par NetApp. Voir "Prise en charge des versions logicielles".
- NetApp HCI/Element logiciel 11 ou supérieur

Prise en charge de Trident pour KubeVirt et la virtualisation OpenShift

Pilotes de stockage pris en charge :

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

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

Points à prendre en considération :

- Mettez à jour la classe de stockage pour avoir le `fsType` paramètre (par exemple : `fsType: "ext4"`) dans l'environnement OpenShift Virtualization. Si nécessaire, définissez explicitement le mode de volume sur bloc à l'aide du `volumeMode=Block` paramètre dans le `dataVolumeTemplates` pour notifier à CDI de créer des volumes de données de type Block.
- *Mode d'accès RWX pour les pilotes de stockage bloc* : les pilotes ontap-san (iSCSI, NVMe/TCP, FC) et ontap-san-economy (iSCSI) ne sont pris en charge qu'avec « volumeMode: Block » (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 à chaud 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)

Exigences fonctionnelles

Le tableau ci-dessous récapitule les fonctionnalités disponibles avec cette version de Trident et les versions de Kubernetes qu'elle prend en charge.

Fonctionnalité	Version Kubernetes	Portails fonctionnels requis ?
Trident	1.27 - 1.36	Non
Instantanés de volume	1.27 - 1.36	Non
PVC à partir de Volume Snapshots	1.27 - 1.36	Non
Redimensionnement iSCSI PV	1.27 - 1.36	Non
ONTAP CHAP bidirectionnel	1.27 - 1.36	Non
Politiques 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 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 nécessite 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'exécutera donc sur n'importe quel worker Linux. Cependant, ces workers doivent pouvoir monter les volumes que Trident fournit à l'aide du client NFS standard ou de l'initiateur iSCSI, selon les backends que vous utilisez.

L' `tridentctl`utilitaire fonctionne également sur toutes ces distributions de Linux.

Configuration de l'hôte

Tous les nœuds de travail du cluster Kubernetes doivent pouvoir monter les volumes que vous avez provisionnés pour vos pods. Pour préparer les nœuds de travail, vous devez installer les outils NFS, iSCSI ou NVMe en fonction de votre sélection de pilote.

"Préparez le nœud de travail"

Configuration du système de stockage

Trident peut nécessiter des modifications d'un système de stockage avant qu'une configuration backend puisse l'utiliser.

"Configurer les backends"

Ports Trident

Trident nécessite l'accès à des ports spécifiques pour communiquer.

"Ports Trident"

Images de conteneur et versions Kubernetes correspondantes

Pour les installations isolées du réseau, la liste suivante répertorie les images de conteneurs nécessaires à l'installation de Trident. Utilisez la `tridentctl images` commande pour vérifier la liste des images de conteneurs 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, v1.36.0	• 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)

Installer Trident

Découvrez l'installation de Trident

Pour garantir que Trident puisse être installé dans une grande variété d'environnements et d'organisations, NetApp propose plusieurs options d'installation. Vous pouvez installer Trident à l'aide de l'opérateur Trident (manuellement ou avec Helm) ou avec `tridentctl`. Cette rubrique fournit des informations importantes pour choisir le bon processus d'installation pour vous.

Informations essentielles concernant Trident 26.02

Vous devez lire les informations essentielles suivantes concernant Trident.

Informations essentielles sur Trident

- Kubernetes 1.35 est désormais pris en charge dans Trident. Mettez à niveau Trident avant de mettre à niveau Kubernetes.
- Trident impose strictement l'utilisation de la configuration multipath dans les environnements SAN, avec une valeur recommandée de `find_multipaths: no` dans le fichier multipath.conf.

  L'utilisation d'une configuration sans multipath ou l'utilisation de `find_multipaths: yes` ou `find_multipaths: smart` dans le 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 "exigences" pour plus de détails.
- Accès à un système de stockage NetApp pris en charge.
- Possibilité 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.
- La variable d'environnement `KUBECONFIG` définie pour pointer vers la configuration de votre cluster Kubernetes.
- Si vous utilisez Kubernetes avec Docker Enterprise "Suivez leurs étapes pour activer l'accès à l'interface de ligne de commande".
- Le cluster doit prendre en charge les charges de travail privilégiées.

Si vous ne vous êtes pas encore 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 "passer 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 "personnaliser le déploiement de votre opérateur Trident" en utilisant les attributs de la `TridentOrchestrator` ressource personnalisée (CR).

Les avantages de l'utilisation de l'opérateur Trident incluent :

Création d'objet Trident

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

- ServiceAccount pour l'opérateur
- ClusterRole et ClusterRoleBinding à la ServiceAccount
- Dédié PodSecurityPolicy (pour Kubernetes 1.25 et versions antérieures)
- L'opérateur lui-même

Responsabilité des ressources

L'opérateur Trident à l'échelle du cluster gère les ressources associées à une installation Trident au niveau du cluster. Cela permet d'éviter les erreurs pouvant survenir lors de la gestion des ressources à l'échelle du cluster à l'aide d'un opérateur à l'échelle de l'espace de noms. Ceci est essentiel pour l'autoréparation et l'application de correctifs.

Capacité d'auto-guérison

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

Mises à jour faciles des installations existantes

Vous pouvez facilement mettre à jour un déploiement existant avec l'opérateur. Vous devez simplement modifier le `TridentOrchestrator` CR pour effectuer des mises à jour sur une installation.

Par exemple, imaginez un scénario où vous devez activer Trident pour générer des journaux de débogage. Pour ce faire, modifiez votre `TridentOrchestrator` pour définir `spec.debug` sur `true` :

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

Après `TridentOrchestrator` la mise à jour, l'opérateur traite les mises à jour et applique des correctifs à l'installation existante. Cela peut déclencher la création de nouveaux pods pour modifier l'installation en conséquence.

Réinstallation propre

L'opérateur Trident à l'échelle du cluster permet la suppression propre des ressources à l'échelle du cluster. Les utilisateurs peuvent désinstaller complètement Trident et le réinstaller facilement.

Gestion automatique de la mise à niveau Kubernetes

Lorsque la version Kubernetes du cluster est mise à niveau vers une version prise en charge, l'opérateur met automatiquement à jour l'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 de Kubernetes non prise en charge.

En utilisant `tridentctl`

Si vous disposez d'un déploiement existant qui doit être mis à niveau ou si vous souhaitez personnaliser fortement votre déploiement, vous devriez envisager . Il s'agit de la méthode conventionnelle de déploiement de Trident.

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

> À compter 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 persiste d'une installation à l'autre. Cela signifie que, `tridentctl` dans la version 22.04, vous pouvez désinstaller les versions précédentes de Trident, mais les versions antérieures ne peuvent pas désinstaller les installations 22.04. Sélectionnez la méthode d'installation appropriée.

Choisissez votre mode d'installation

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

Installation standard

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

Lorsque vous utilisez le mode standard, l'installateur Trident :

- Récupère les images des conteneurs sur Internet
- Crée un déploiement ou un daemonset de nœud, qui lance des pods Trident sur tous les nœuds éligibles du cluster Kubernetes

Installation hors ligne

L'installation hors ligne peut être nécessaire dans un environnement isolé ou sécurisé. Dans ce cas, vous pouvez créer un registre privé unique 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 seul registre.

Installation à distance

Voici un aperçu général 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 définissez la variable d'environnement `KUBECONFIG` sur la machine distante.
- Lancez une commande `kubectl get nodes` pour vérifier que vous pouvez vous connecter au cluster Kubernetes requis.
- Finalisez le déploiement depuis la machine distante en utilisant 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 (manuel)	"Installation standard" "Installation hors ligne"
Opérateur Trident (Helm)	"Installation standard" "Installation hors ligne"

Méthode	Mode d'installation
`tridentctl`	"Installation standard ou hors ligne"

Passer d'une méthode d'installation à une autre

Vous pouvez choisir de modifier votre méthode d'installation. Avant de le faire, tenez compte des 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 binaire `tridentctl` pour désinstaller Trident. De même, si vous déployez avec l'opérateur, vous devez modifier le CR `TridentOrchestrator` et définir `spec.uninstall=true` pour désinstaller Trident.
- Si vous disposez d'un déploiement basé sur un 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` pour désinstaller Trident. Supprimez ensuite `TridentOrchestrator` et le déploiement de l'opérateur. Vous pouvez ensuite installer en utilisant `tridentctl`.
- Si vous avez un déploiement manuel basé sur un opérateur et que vous souhaitez utiliser un déploiement de l'opérateur Trident basé sur Helm, vous devez d'abord désinstaller manuellement l'opérateur, puis effectuer l'installation Helm. Cela permet à Helm de déployer l'opérateur Trident avec les étiquettes et annotations requises. Si vous ne le faites pas, votre déploiement de l'opérateur Trident basé sur Helm échouera avec une erreur de validation des étiquettes et une erreur de validation des annotations.
- Si vous disposez d'un déploiement basé sur `tridentctl`, 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 VMWare Tanzu Portfolio :

- Le `--kubelet-dir` paramètre doit être défini sur l'emplacement du répertoire kubelet. Par défaut, c'est `/var/vcap/data/kubelet`.

  La spécification de l'emplacement du kubelet à l'aide de `--kubelet-dir` est connue pour fonctionner avec Trident Operator, Helm et les déploiements `tridentctl`.

Installer à l'aide de l'opérateur Trident

Déployez 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 essentielles suivantes concernant Trident.

Informations essentielles sur Trident

- Kubernetes 1.35 est désormais pris en charge dans Trident. Mettez à niveau Trident avant de mettre à niveau Kubernetes.
- Trident impose strictement l'utilisation de la configuration multipath dans les environnements SAN, avec une valeur recommandée de `find_multipaths: no` dans le fichier multipath.conf.

  L'utilisation d'une configuration sans multipath ou l'utilisation de `find_multipaths: yes` ou `find_multipaths: smart` dans le fichier multipath.conf entraînera des échecs de montage. Trident a recommandé l'utilisation de `find_multipaths: no` depuis la version 21.07.

Déployez manuellement l'opérateur Trident et installez Trident

Vérifiez "aperçu 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 d'abord en tant que **system:admin** en exécutant `oc login -u system:admin` ou `oc login -u kube-admin`.

1. Vérifiez votre version de Kubernetes :

```
kubectl version
```

2. Vérifiez les privilèges de l'administrateur du cluster :

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

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

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

Étape 1 : Téléchargez le package d'installation de Trident

Le package d'installation de Trident contient tout le nécessaire pour déployer l'opérateur Trident et installer Trident. Téléchargez et extrayez la dernière version du programme d'installation Trident depuis "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

Créez la `TridentOrchestrator` définition de ressource personnalisée (CRD). Vous créerez une `TridentOrchestrator` ressource personnalisée ultérieurement. Utilisez la version YAML CRD appropriée dans `deploy/crds` pour créer la `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 Trident fournit un fichier bundle qui peut être utilisé pour installer l'opérateur et créer les objets associés. Le fichier bundle est un moyen simple de déployer l'opérateur et d'installer Trident en utilisant 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 une version ultérieure, utilisez `bundle_post_1_25.yaml`.

Avant de commencer

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

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

- Pour déployer l'opérateur dans un espace de noms autre que l' `trident` namespace, mettez à jour `serviceaccount.yaml`, `clusterrolebinding.yaml` et `operator.yaml` et générez votre fichier bundle en utilisant le `kustomization.yaml`.

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

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

  b. Compilez le bundle en utilisant la commande suivante où *<bundle.yaml>* est `bundle_pre_1_25.yaml` ou `bundle_post_1_25.yaml` selon 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 replicasets ont été créés.

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

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

Étape 4 : Créez le `TridentOrchestrator` et installez Trident

Vous pouvez maintenant créer le `TridentOrchestrator` et installer Trident. Vous pouvez également "Personnalisez votre installation Trident" en utilisant les attributs dans la spécification `TridentOrchestrator`.

```
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.

Utilisation `TridentOrchestrator` du statut

L'état de `TridentOrchestrator` indique si l'installation a réussi et affiche la version de Trident installée. Pendant l'installation, l'état de `TridentOrchestrator` change de `Installing` à `Installed`. Si vous observez l'état `Failed` et que l'opérateur ne parvient pas à se rétablir tout seul, "vérifiez les journaux".

Statut	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, parce que `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 se rétablir de cet état. Si cet état persiste, un dépannage sera nécessaire.
Mise à jour	L'opérateur met à jour une installation existante.
Erreur	Le `TridentOrchestrator` n'est pas utilisé. Un autre existe déjà.

Utilisation de l'état de création du pod

Vous pouvez confirmer si l'installation de Trident est terminée en consultant le statut des pods créés :

```
kubectl get pods -n trident

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

En utilisant `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        |
+----------------+----------------+
```

Déployez manuellement 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 "processus de déploiement standard".

Informations essentielles sur Trident

Vous devez lire les informations essentielles suivantes concernant Trident.

Informations essentielles sur Trident

- Kubernetes 1.35 est désormais pris en charge dans Trident. Mettez à niveau Trident avant de mettre à niveau Kubernetes.
- Trident impose strictement l'utilisation de la configuration multipath dans les environnements SAN, avec une valeur recommandée de `find_multipaths: no` dans le fichier multipath.conf.

  L'utilisation d'une configuration sans multipath ou l'utilisation de `find_multipaths: yes` ou `find_multipaths: smart` dans le fichier multipath.conf entraînera des échecs de montage. Trident a recommandé l'utilisation de `find_multipaths: no` depuis la version 21.07.

Déployez manuellement l'opérateur Trident et installez Trident

Vérifiez "aperçu 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 d'abord en tant que **system:admin** en exécutant `oc login -u system:admin` ou `oc login -u kube-admin`.

1. Vérifiez votre version de Kubernetes :

```
kubectl version
```

2. Vérifiez les privilèges de l'administrateur du cluster :

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

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

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

Étape 1 : Téléchargez le package d'installation de Trident

Le package d'installation de Trident contient tout le nécessaire pour déployer l'opérateur Trident et installer Trident. Téléchargez et extrayez la dernière version du programme d'installation Trident depuis "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

Créez la `TridentOrchestrator` définition de ressource personnalisée (CRD). Vous créerez une `TridentOrchestrator` ressource personnalisée ultérieurement. Utilisez la version YAML CRD appropriée dans `deploy/crds` pour créer la `TridentOrchestrator` CRD :

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

Étape 3 : Mettre 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 de Trident et de 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 : Déployer l'opérateur Trident

Le programme d'installation Trident fournit un fichier bundle qui peut être utilisé pour installer l'opérateur et créer les objets associés. Le fichier bundle est un moyen simple de déployer l'opérateur et d'installer Trident en utilisant 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 une version ultérieure, utilisez `bundle_post_1_25.yaml`.

Avant de commencer

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

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

- Pour déployer l'opérateur dans un espace de noms autre que l' `trident` namespace, mettez à jour `serviceaccount.yaml`, `clusterrolebinding.yaml` et `operator.yaml` et générez votre fichier bundle en utilisant le `kustomization.yaml`.

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

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

  b. Compilez le bundle en utilisant la commande suivante où *<bundle.yaml>* est `bundle_pre_1_25.yaml` ou `bundle_post_1_25.yaml` selon 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 replicasets ont été créés.

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

Il ne doit y avoir qu'une seule 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 de Trident et de CSI" peut être situé dans un registre ou dans des registres différents, mais toutes les images CSI doivent être situées dans le même registre. Mettez à jour `deploy/crds/tridentorchestrator_cr.yaml` pour ajouter les spécifications d'emplacement supplémentaires en fonction de la configuration de votre registre.

Images dans un seul 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 installez Trident

Vous pouvez maintenant créer le `TridentOrchestrator` et installer Trident. Vous pouvez également "Personnalisez votre installation Trident"affiner l'installation à l'aide des attributs dans la `TridentOrchestrator` spécification. L'exemple suivant montre une installation où les images Trident et CSI sont situées dans des registres différents.

```
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.

Utilisation `TridentOrchestrator` du statut

L'état de `TridentOrchestrator` indique si l'installation a réussi et affiche la version de Trident installée. Pendant l'installation, l'état de `TridentOrchestrator` change de `Installing` à `Installed`. Si vous observez l'état `Failed` et que l'opérateur ne parvient pas à se rétablir tout seul, "vérifiez les journaux".

Statut	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, parce que `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 se rétablir de cet état. Si cet état persiste, un dépannage sera nécessaire.
Mise à jour	L'opérateur met à jour une installation existante.
Erreur	Le `TridentOrchestrator` n'est pas utilisé. Un autre existe déjà.

Utilisation de l'état de création du pod

Vous pouvez confirmer si l'installation de Trident est terminée en consultant le statut des pods créés :

```
kubectl get pods -n trident

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

En utilisant `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        |
+----------------+----------------+
```

Déployez 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 Trident 25.10

Vous devez lire les informations essentielles suivantes concernant Trident.

Informations essentielles sur Trident

- Kubernetes 1.35 est désormais pris en charge dans Trident. Mettez à niveau Trident avant de mettre à niveau Kubernetes.
- Trident impose strictement l'utilisation de la configuration multipath dans les environnements SAN, avec une valeur recommandée de `find_multipaths: no` dans le fichier multipath.conf.

  L'utilisation d'une configuration sans multipath ou l'utilisation de `find_multipaths: yes` ou `find_multipaths: smart` dans le 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 de Helm

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

Vérifiez "aperçu 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 "prérequis de déploiement" vous avez besoin de "Helm version 3".

Étapes

1. Ajoutez le dépôt 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 espace de noms pour Trident, le `--create-namespace` paramètre ne créera pas d'espace de noms supplémentaire.

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

Transmettez les données de configuration lors de l'installation

Il existe deux façons de transmettre les données de configuration lors de l'installation :

Option	Description
`--values` (ou `-f`)	Spécifiez un fichier YAML contenant les règles de remplacement. Vous pouvez le spécifier plusieurs fois et le fichier le plus à droite sera prioritaire.
`--set`	Spécifiez les substitutions sur la ligne de commandes.

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` fichier, qui fait partie du Helm chart, fournissent la liste des clés et leurs valeurs par défaut.

Option	Description	Défaut
`nodeSelector`	Étiquettes de nœud pour l'affectation des pods	
`podAnnotations`	Annotations des pods	

Option	Description	Défaut
`deploymentAnnotations`	Annotations de déploiement	
`tolerations`	Tolérances pour l'affectation des pods	

`affinity`	Affinité pour l'affectation des pods	`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. Consultez Comprendre les pods de contrôleur et les pods de nœud pour plus de détails.	
`tridentControllerPluginTolerations`	Remplace les tolérances Kubernetes pour les pods. Consultez Comprendre les pods de contrôleur et les pods de nœud pour plus de détails.	
`tridentNodePluginNodeSelector`	Sélecteurs de nœuds supplémentaires pour les pods. Consultez Comprendre les pods de contrôleur et les pods de nœud pour plus de détails.	

Option	Description	Défaut
`tridentNodePluginTolerations`	Remplace les tolérances Kubernetes pour les pods. Consultez Comprendre les pods de contrôleur et les pods de nœud pour plus de détails.	
`imageRegistry`	Identifie le registre pour les `trident-operator`, `trident` et d'autres images. Laissez vide pour accepter la valeur par défaut. IMPORTANT : Lors de l'installation de Trident dans un dépôt privé, si vous utilisez le commutateur `imageRegistry` pour spécifier l'emplacement du dépôt, n'utilisez pas `/netapp/` dans le chemin du dépôt.	`""`
`imagePullPolicy`	Définit la politique d'extraction d'images pour le `trident-operator`.	`IfNotPresent`
`imagePullSecrets`	Définit les secrets d'extraction d'images pour les `trident-operator`, `trident` et d'autres images.	
`kubeletDir`	Permet de remplacer l'emplacement hôte de l'état interne de kubelet.	`"/var/lib/kubelet"`
`operatorLogLevel`	Permet de définir le niveau de journalisation de l'opérateur Trident sur : `trace`, `debug`, `info`, `warn`, `error` ou `fatal`.	`"info"`
`operatorDebug`	Permet de définir le niveau de journalisation de l'opérateur Trident sur débogage.	`true`
`operatorImage`	Permet de remplacer complètement l'image pour `trident-operator`.	`""`
`operatorImageTag`	Permet de remplacer l'étiquette de l' `trident-operator`image.	`""`
`tridentIPv6`	Permet d'activer Trident pour fonctionner dans des clusters IPv6.	`false`
`tridentK8sTimeout`	Remplace le délai d'expiration par défaut de 30 secondes pour la plupart des opérations de l'API Kubernetes (si non nul, en secondes).	`0`

Option	Description	Défaut
`tridentHttpRequestTimeout`	Permet de remplacer le délai d'expiration par défaut de 90 secondes pour les requêtes HTTP, `0s` étant une durée infinie pour le délai d'expiration. Les valeurs négatives ne sont pas autorisées.	`"90s"`
`tridentSilenceAutosupport`	Permet de désactiver les rapports périodiques AutoSupport de Trident.	`false`
`tridentAutosupportImageTag`	Permet de remplacer l'étiquette de l'image pour le conteneur Trident AutoSupport.	`<version>`
`tridentAutosupportProxy`	Permet au conteneur Trident AutoSupport de communiquer avec son serveur via un proxy HTTP.	`""`
`tridentLogFormat`	Définit le format de journalisation Trident (`text` ou `json`).	`"text"`
`tridentDisableAuditLog`	Désactive le journal d'audit Trident.	`true`
`tridentLogLevel`	Permet de définir le niveau de journalisation de Trident sur : `trace`, `debug`, `info`, `warn`, `error` ou `fatal`.	`"info"`
`tridentDebug`	Permet de définir le niveau de journalisation de Trident sur `debug`. Vous pouvez automatiser le processus de déconnexion forcée grâce à l'intégration avec l'opérateur de vérification de l'état du nœud (NHC). Pour plus d'informations, consultez "Automatisation du basculement des applications avec état avec Trident".	`false`
`tridentLogWorkflows`	Permet d'activer des flux de travail Trident spécifiques pour la journalisation des traces ou la suppression des journaux.	`""`
`tridentLogLayers`	Permet d'activer des couches Trident spécifiques pour la journalisation des traces ou la suppression des journaux.	`""`
`tridentImage`	Permet de remplacer complètement l'image pour Trident.	`""`

Option	Description	Défaut
`tridentImageTag`	Permet de remplacer l'étiquette de l'image pour Trident.	`""`
`tridentProbePort`	Permet de remplacer le port par défaut utilisé pour les sondes de liveness/readiness de Kubernetes.	`""`
`windows`	Permet d'installer Trident sur un nœud de travail Windows.	`false`
`enableForceDetach`	Permet d'activer la fonction de détachement forcé.	`false`
`excludePodSecurityPolicy`	Exclut la création de la politique de sécurité du pod opérateur.	`false`
`cloudProvider`	Définissez sur `"Azure"` lors de l'utilisation d'identités gérées ou d'une identité cloud sur un cluster AKS. Définissez sur "AWS" lors de l'utilisation d'une identité cloud sur un cluster EKS.	`""`
`cloudIdentity`	Définissez sur workload identity (« azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxx ») lors de l'utilisation de l'identité cloud sur un cluster AKS. Définissez sur AWS IAM role (« '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 auquel l'auto-réparation iSCSI est déclenchée.	`5m0s`
`iscsiSelfHealingWaitTime`	La durée après laquelle l'auto-réparation iSCSI initie une tentative de résolution d'une session obsolète en effectuant une déconnexion puis une reconnexion.	`7m0s`
`nodePrep`	Permet à Trident de préparer les nœuds du cluster Kubernetes à gérer les volumes en utilisant le protocole de stockage des données spécifié. **Actuellement, `iscsi` est la seule valeur prise en charge.** REMARQUE : À partir de OpenShift 4.19, la version minimale de Trident prise en charge pour cette fonctionnalité est 25.06.1.	

Option	Description	Défaut
`enableConcurrency`	Permet les 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 unified ONTAP 9), en plus de l'aperçu technique existant pour le pilote ONTAP-SAN (protocoles iSCSI et FCP dans unified ONTAP 9).	false
`k8sAPIQPS`	Limite de requêtes par seconde (QPS) utilisée par le contrôleur lors de la communication avec le serveur d'API Kubernetes. La valeur de Burst est définie automatiquement en fonction de la valeur QPS.	`100`; facultatif

Option	Description	Défaut

`resources`	Définit les limites et les demandes 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 afin de gérer l'allocation des ressources dans Kubernetes.
	
	Pour plus d'informations sur la configuration des demandes et des limites de ressources, consultez "Gestion des ressources pour les pods et les conteneurs".
	
	• NE MODIFIEZ PAS les noms de conteneurs ou de 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	Défaut
`httpsMetrics`	Activer HTTPS pour le point de terminaison des métriques Prometheus.	false
`hostNetwork`	Active la mise en réseau de l'hôte pour le contrôleur Trident. Ceci est utile lorsque vous souhaitez séparer le trafic frontal et dorsal dans un réseau multi-home.	false

Comprendre les pods de contrôleur et les pods de nœud

Trident s'exécute sous la forme d'un pod de contrôleur unique, ainsi que d'un pod de nœud sur chaque nœud de travail du cluster. Le pod de nœud doit être exécuté sur tout hôte sur lequel vous souhaitez potentiellement monter un volume Trident.

Kubernetes "sélecteurs de nœuds" et "tolérances et taints" 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 substitutions.

- Le plugin de contrôleur gère le provisionnement et la gestion des volumes, comme les instantanés et le redimensionnement.
- Le plugin de nœud gère la connexion du stockage au nœud.

Déployez 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 "processus de déploiement standard".

Informations essentielles concernant Trident 26.02

Vous devez lire les informations essentielles suivantes concernant Trident.

Informations essentielles sur Trident

- Kubernetes 1.35 est désormais pris en charge dans Trident. Mettez à niveau Trident avant de mettre à niveau Kubernetes.
- Trident impose strictement l'utilisation de la configuration multipath dans les environnements SAN, avec une valeur recommandée de `find_multipaths: no` dans le fichier multipath.conf.

  L'utilisation d'une configuration sans multipath ou l'utilisation de `find_multipaths: yes` ou `find_multipaths: smart` dans le 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 de Helm

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

Vérifiez "aperçu 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 "prérequis de déploiement" vous avez besoin de "Helm version 3".

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

Étapes

1. Ajoutez le dépôt 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 de Trident et de 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 seul 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 espace de noms pour Trident, le `--create-namespace` paramètre ne créera pas d'espace de noms supplémentaire.

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

Transmettez les données de configuration lors de l'installation

Il existe deux façons de transmettre les données de configuration lors de l'installation :

Option	Description
`--values` (ou `-f`)	Spécifiez un fichier YAML contenant les règles de remplacement. Vous pouvez le spécifier plusieurs fois et le fichier le plus à droite sera prioritaire.
`--set`	Spécifiez les substitutions sur la ligne de commandes.

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` fichier, qui fait partie du Helm chart, fournissent 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	Défaut
`nodeSelector`	Étiquettes de nœud pour l'affectation des pods	
`podAnnotations`	Annotations des pods	
`deploymentAnnotations`	Annotations de déploiement	
`tolerations`	Tolérances pour l'affectation des pods	

Option	Description	Défaut

`affinity`	Affinité pour l'affectation des pods	`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.

`tridentControllerPluginNodeSelector`	Sélecteurs de nœuds supplémentaires pour les pods. Consultez "Comprendre les pods de contrôleur et les pods de nœud" pour plus de détails.	
`tridentControllerPluginTolerations`	Remplace les tolérances Kubernetes pour les pods. Consultez "Comprendre les pods de contrôleur et les pods de nœud" pour plus de détails.	

Option	Description	Défaut
`tridentNodePluginNodeSelector`	Sélecteurs de nœuds supplémentaires pour les pods. Consultez "Comprendre les pods de contrôleur et les pods de nœud" pour plus de détails.	
`tridentNodePluginTolerations`	Remplace les tolérances Kubernetes pour les pods. Consultez "Comprendre les pods de contrôleur et les pods de nœud" pour plus de détails.	
`imageRegistry`	Identifie le registre pour les `trident-operator`, `trident` et d'autres images. Laissez vide pour accepter la valeur par défaut. IMPORTANT : Lors de l'installation de Trident dans un dépôt privé, si vous utilisez le commutateur `imageRegistry` pour spécifier l'emplacement du dépôt, n'utilisez pas `/netapp/` dans le chemin du dépôt.	""
`imagePullPolicy`	Définit la politique d'extraction d'images pour le `trident-operator`.	`IfNotPresent`
`imagePullSecrets`	Définit les secrets d'extraction d'images pour les `trident-operator`, `trident` et d'autres images.	
`kubeletDir`	Permet de remplacer l'emplacement hôte de l'état interne de kubelet.	`"/var/lib/kubelet"`
`operatorLogLevel`	Permet de définir le niveau de journalisation de l'opérateur Trident sur : `trace`, `debug`, `info`, `warn`, `error` ou `fatal`.	`"info"`
`operatorDebug`	Permet de définir le niveau de journalisation de l'opérateur Trident sur débogage.	`true`
`operatorImage`	Permet de remplacer complètement l'image pour `trident-operator`.	""
`operatorImageTag`	Permet de remplacer l'étiquette de l' `trident-operator`image.	""
`tridentIPv6`	Permet d'activer Trident pour fonctionner dans des clusters IPv6.	`false`

Option	Description	Défaut
`tridentK8sTimeout`	Remplace le délai d'expiration par défaut de 180 secondes pour la plupart des opérations de l'API Kubernetes (si non nul, en secondes). Le `tridentK8sTimeout` paramètre s'applique uniquement à l'installation de Trident.	`180`
`tridentHttpRequestTimeout`	Permet de remplacer le délai d'expiration par défaut de 90 secondes pour les requêtes HTTP, `0s` étant une durée infinie pour le délai d'expiration. Les valeurs négatives ne sont pas autorisées.	`"90s"`
`tridentSilenceAutosupport`	Permet de désactiver les rapports périodiques AutoSupport de Trident.	`false`
`tridentAutosupportImageTag`	Permet de remplacer l'étiquette de l'image pour le conteneur Trident AutoSupport.	`<version>`
`tridentAutosupportProxy`	Permet au conteneur Trident AutoSupport de communiquer avec son serveur via un proxy HTTP.	""
`tridentLogFormat`	Définit le format de journalisation Trident (`text` ou `json`).	`"text"`
`tridentDisableAuditLog`	Désactive le journal d'audit Trident.	`true`
`tridentLogLevel`	Permet de définir le niveau de journalisation de Trident sur : `trace`, `debug`, `info`, `warn`, `error` ou `fatal`.	`"info"`
`tridentDebug`	Permet de définir le niveau de journalisation de Trident sur `debug`.	`false`
`tridentLogWorkflows`	Permet d'activer des flux de travail Trident spécifiques pour la journalisation des traces ou la suppression des journaux.	""
`tridentLogLayers`	Permet d'activer des couches Trident spécifiques pour la journalisation des traces ou la suppression des journaux.	""

Option	Description	Défaut
`tridentImage`	Permet de remplacer complètement l'image pour Trident.	""
`tridentImageTag`	Permet de remplacer l'étiquette de l'image pour Trident.	""
`tridentProbePort`	Permet de remplacer le port par défaut utilisé pour les sondes de liveness/readiness de Kubernetes.	""
`windows`	Permet d'installer Trident sur un nœud de travail Windows.	`false`
`enableForceDetach`	Permet d'activer la fonction de détachement forcé. Vous pouvez automatiser le processus de déconnexion forcée grâce à l'intégration avec l'opérateur de vérification de l'état du nœud (NHC). Pour plus d'informations, consultez "Automatisation du basculement des applications avec état avec Trident".	`false`
`excludePodSecurityPolicy`	Exclut la création de la politique de sécurité du pod opérateur.	`false`
`nodePrep`	Permet à Trident de préparer les nœuds du cluster Kubernetes à gérer les volumes en utilisant le protocole de stockage des données spécifié. **Actuellement, `iscsi` est la seule valeur prise en charge.** À partir de OpenShift 4.19, la version minimale de Trident prise en charge pour cette fonctionnalité est 25.06.1.	

Option	Description	Défaut

`resources`	Définit les limites et les demandes 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 afin de gérer l'allocation des ressources dans Kubernetes.
	
	Pour plus d'informations sur la configuration des demandes et des limites de ressources, consultez "Gestion des ressources pour les pods et les conteneurs".
	
	• NE MODIFIEZ PAS les noms de conteneurs ou de 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:`

Personnaliser l'installation de l'opérateur Trident

L'opérateur Trident vous permet de personnaliser l'installation de Trident à l'aide des attributs dans la `TridentOrchestrator`spec. Si vous souhaitez personnaliser l'installation au-delà de ce que les arguments `TridentOrchestrator`permettent, envisagez d'utiliser `tridentctl`pour générer des manifestes YAML personnalisés à modifier selon vos besoins.

Comprendre les pods de contrôleur et les pods de nœud

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 de nœud doit être exécuté sur tout hôte sur lequel vous souhaitez potentiellement monter un volume Trident.

Kubernetes "sélecteurs de nœuds" et "tolérances et taints" 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 substitutions.

- Le plugin de contrôleur gère le provisionnement et la gestion des volumes, comme les instantanés et le redimensionnement.
- Le plugin de nœud gère la connexion du stockage au nœud.

Options de configuration

> `spec.namespace` est spécifié dans `TridentOrchestrator` pour indiquer l'espace de noms où Trident est installé. Ce paramètre **ne peut pas être mis à jour après l'installation de Trident**. Tenter de le faire provoque le changement du statut `TridentOrchestrator` en `Failed`. Trident n'est pas destiné à être migré entre des espaces de noms.

Ce tableau détaille `TridentOrchestrator` les attributs.

Paramètre	Description	Défaut
`namespace`	Espace de noms dans lequel installer Trident	`"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, consultez "Automatisation du basculement des applications avec état avec Trident".	`false`
`windows`	Le paramètre `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 sidecars sont répertoriés sans distinction conteneur principal.
- Consultez le champ `status.CurrentInstallationParams` TORC pour afficher les valeurs actuellement appliquées.

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

Paramètre	Description	Défaut
`cloudProvider`	Définissez sur `"Azure"` lors de l'utilisation d'identités gérées ou d'une identité cloud sur un cluster AKS. Définir sur `"AWS"` lors de l'utilisation d'une identité cloud sur un cluster EKS. Définir sur `"GCP"` lors de l'utilisation d'une identité cloud sur un cluster GKE.	`""`
`cloudIdentity`	Définissez sur workload identity (« azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxx ») lors de l'utilisation de l'identité cloud sur un cluster AKS. Définissez sur AWS IAM role (« 'eks.amazonaws.com/role-arn: arn:aws:iam::123456:role/trident-role' ») lors de l'utilisation de l'identité cloud sur un cluster EKS. Définissez sur cloud identity ("'iam.gke.io/gcp-service-account: xxxx@mygcpproject.iam.gserviceaccount.com'") lors de l'utilisation de l'identité cloud sur un cluster GKE.	`""`
`IPv6`	Installer Trident sur IPv6	false
`k8sTimeout`	Délai d'attente pour les opérations Kubernetes. Le `k8sTimeout`paramètre s'applique uniquement à l'installation de Trident.	`180sec`
`silenceAutosupport`	N'envoyez pas automatiquement les bundles d'autosupport à NetApp	`false`
`autosupportImage`	L'image conteneur pour la télémétrie Autosupport	`"netapp/trident-autosupport10"`
`autosupportProxy`	L'adresse/port d'un proxy pour l'envoi des données de télémétrie Autosupport	`"http://proxy.example.com:8888"`
`uninstall`	Un 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 sur l'hôte	`"/var/lib/kubelet"`
`wipeout`	Une liste des ressources à supprimer pour effectuer une suppression complète de Trident	
`imagePullSecrets`	Secrets pour extraire des images d'un registre interne	

Paramètre	Description	Défaut
`imagePullPolicy`	Définit la stratégie de récupération d'images pour l'opérateur Trident. Les valeurs valides sont : `Always` pour toujours récupérer l'image. `IfNotPresent` pour récupérer l'image uniquement si elle n'existe pas déjà sur le nœud. `Never` pour ne jamais récupérer l'image.	`IfNotPresent`
`controllerPluginNodeSelector`	Sélecteurs de nœuds supplémentaires pour les pods. Suit le même format que `pod.spec.nodeSelector`.	Aucune valeur par défaut ; optionnel
`controllerPluginTolerations`	Remplace les tolérances Kubernetes pour les pods. Suit le même format que `pod.spec.Tolerations`.	Aucune valeur par défaut ; optionnel
`nodePluginNodeSelector`	Sélecteurs de nœuds supplémentaires pour les pods. Suit le même format que `pod.spec.nodeSelector`.	Aucune valeur par défaut ; optionnel
`nodePluginTolerations`	Remplace les tolérances Kubernetes pour les pods. Suit le même format que `pod.spec.Tolerations`.	Aucune valeur par défaut ; optionnel
`nodePrep`	Permet à Trident de préparer les nœuds du cluster Kubernetes à gérer les volumes en utilisant le protocole de stockage des données spécifié. **Actuellement, `iscsi` est la seule valeur prise en charge.** ⓘ À partir de OpenShift 4.19, la version minimale de Trident prise en charge pour cette fonctionnalité est 25.06.1.	
`k8sAPIQPS`	Limite de requêtes par seconde (QPS) utilisée par le contrôleur lors de la communication avec le serveur d'API Kubernetes. La valeur de Burst est définie automatiquement en fonction de la valeur QPS.	`100`; facultatif
`enableConcurrency`	Permet les 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 unified ONTAP 9), en plus de l'aperçu technique existant pour le pilote ONTAP-SAN (protocoles iSCSI et FCP dans unified ONTAP 9).	false

Paramètre	Description	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 afin de gérer l'allocation des ressources dans Kubernetes.
	
	Pour plus d'informations sur la configuration des demandes et des limites de ressources, consultez "Gestion des ressources pour les pods et les conteneurs".
	
	• NE MODIFIEZ PAS les noms de conteneurs ou de 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 sidecars sont répertoriés sous chaque conteneur principal.
	• Consultez le champ `status.CurrentInstallationParams` TORC pour 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	Défaut
`httpsMetrics`	Activer HTTPS pour le point de terminaison des métriques Prometheus.	false
`hostNetwork`	Active la mise en réseau de l'hôte pour le contrôleur Trident. Ceci est utile lorsque vous souhaitez séparer le trafic frontal et dorsal dans un réseau multi-home.	false

Pour plus d'informations sur la mise en forme des paramètres de pod, consultez "Affectation des pods aux nœuds".

Exemples de configurations

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

Configuration personnalisée de base

Cet exemple, créé après l'exécution de la `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
```

Sélecteurs de nœuds

Cet exemple installe Trident avec des sélecteurs de nœuds.

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

Nœuds de travail Windows

Cet exemple, créé après l'exécution de la commande `cat deploy/crds/tridentorchestrator_cr.yaml`, 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
```

Identités gérées sur un cluster AKS

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

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

Identité cloud sur un cluster AKS

Cet exemple installe Trident pour une utilisation avec une identité 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 des 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 données à titre indicatif uniquement. Ajustez ces valeurs en fonction de votre environnement et de vos besoins en 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 données à titre indicatif uniquement. Ajustez ces valeurs en fonction de votre environnement et de vos besoins en 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
```

Installer à l'aide de tridentctl

Installer 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 soit dans un registre privé, soit non. Pour personnaliser votre `tridentctl`déploiement, consultez "Personnalisez le déploiement de tridentctl".

Informations essentielles concernant Trident10

Vous devez lire les informations essentielles suivantes concernant Trident.

Informations essentielles sur Trident

- Kubernetes 1.27 est désormais pris en charge dans Trident. Mettez à niveau Trident avant de mettre à niveau Kubernetes.
- Trident impose strictement l'utilisation de la configuration multipath dans les environnements SAN, avec une valeur recommandée de `find_multipaths: no` dans le fichier multipath.conf.

  L'utilisation d'une configuration sans multipath ou l'utilisation de `find_multipaths: yes` ou `find_multipaths: smart` dans le fichier multipath.conf entraînera des échecs de montage. Trident a recommandé l'utilisation de `find_multipaths: no` depuis la version 21.07.

Installez Trident en utilisant `tridentctl`

Vérifiez "aperçu 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 d'abord en tant que **system:admin** en exécutant `oc login -u system:admin` ou `oc login -u kube-admin`.

1. Vérifiez votre version de Kubernetes :

```
kubectl version
```

2. Vérifiez les privilèges de l'administrateur du cluster :

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

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

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

Étape 1 : Téléchargez le package 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 sidecars CSI pour effectuer des actions telles que le provisionnement et l'attachement de volumes aux hôtes du cluster. Téléchargez et extrayez la dernière version du programme d'installation Trident depuis "la section *Assets* sur GitHub". Mettez à jour *<trident-installer-XX.XX.X.tar.gz>* dans l'exemple avec la version Trident que vous avez 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 : Installer Trident

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

Mode standard

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

Images dans un seul 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 en utilisant l'état de création du pod ou `tridentctl`.

Utilisation de l'état de création du pod

Vous pouvez confirmer si l'installation de Trident est terminée en consultant le statut des pods créés :

```
kubectl get pods -n trident

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

Si 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.

En utilisant `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 configurations pour installer Trident à l'aide de `tridentctl`.

nœuds Windows

Pour permettre à Trident de s'exécuter sur des nœuds Windows :

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

Forcer le détachement

Pour plus d'informations, consultez "Automatisation du basculement des applications avec état avec Trident".

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

Activer les opérations simultanées du contrôleur Trident

Pour activer les opérations simultanées du contrôleur Trident afin d'améliorer le débit, ajoutez l' `--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 unified ONTAP 9), en plus de l'aperçu technique existant pour le pilote ONTAP-SAN (protocoles iSCSI et FCP dans unified ONTAP 9).

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

Personnaliser l'installation de tridentctl

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

Découvrez l'installateur

L'installateur Trident vous permet de personnaliser les attributs. Par exemple, si vous avez copié l'image Trident dans un dépôt privé, vous pouvez spécifier le nom de l'image en utilisant `--trident-image`. Si vous avez copié l'image Trident ainsi que les images CSI sidecar nécessaires dans un dépôt privé, il peut être préférable de spécifier l'emplacement de ce dépôt en utilisant le commutateur `--image-registry`, qui prend la forme `<registry FQDN>[:port]`.

> Lors de l'installation de Trident dans un dépôt privé, si vous utilisez le `--image-registry` commutateur pour spécifier l'emplacement du dépôt, n'utilisez pas `/netapp/` dans le chemin du dépôt. 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 autre que le chemin habituel `/var/lib/kubelet`, vous pouvez spécifier le chemin alternatif en utilisant `--kubelet-dir`.

Si vous devez personnaliser l'installation au-delà des possibilités offertes par les arguments du programme d'installation, vous pouvez également personnaliser les fichiers de déploiement. L'utilisation du paramètre `--generate-custom-yaml` crée les fichiers YAML suivants dans le répertoire `setup` du programme d'installation :

- `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` *

Une fois que vous avez généré ces fichiers, vous pouvez les modifier selon vos besoins, puis utiliser `--use-custom-yaml` pour installer votre déploiement personnalisé.

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

Installer à l'aide de l'opérateur certifié OpenShift

Installez Trident en utilisant OpenShift OperatorHub

Si vous utilisez Red Hat OpenShift, vous pouvez installer NetApp Trident à l'aide de l'opérateur certifié Red Hat. Utilisez cette procédure pour installer Trident depuis la plateforme Red Hat OpenShift Container.

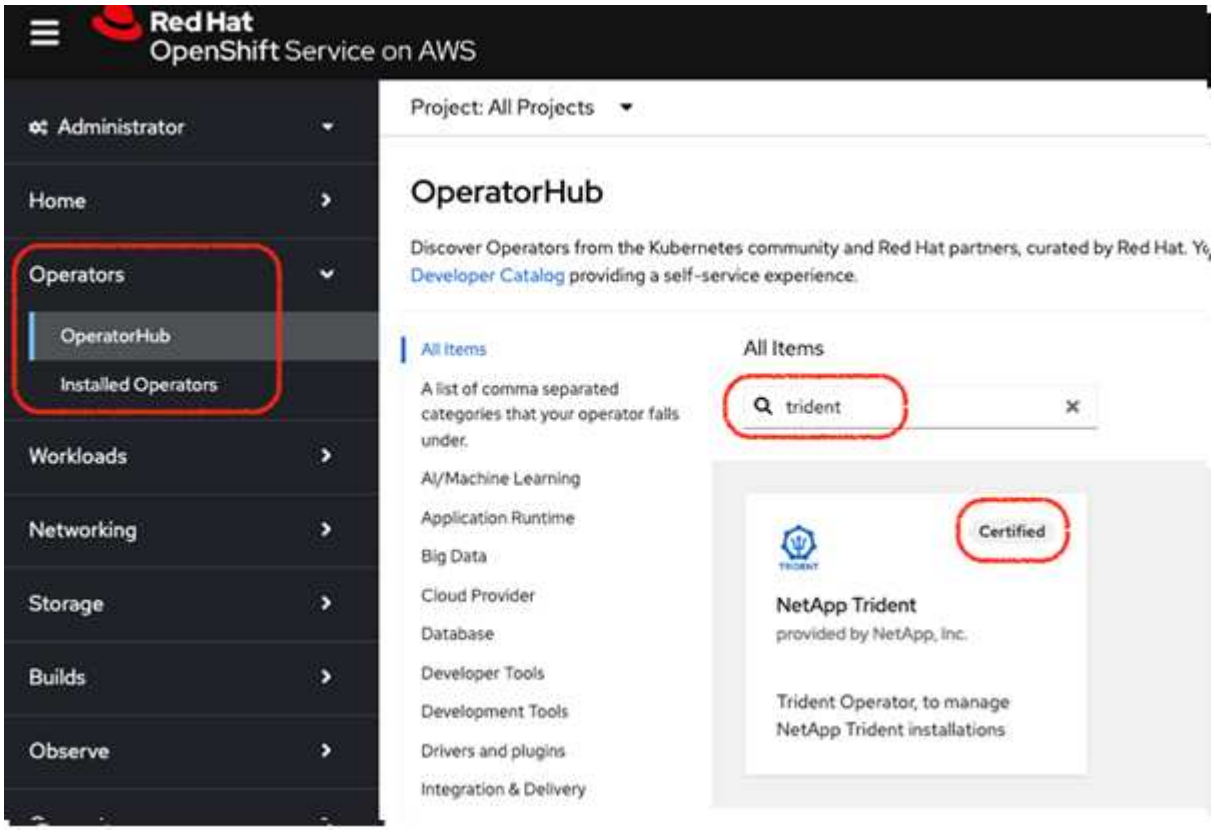
Avant de commencer

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

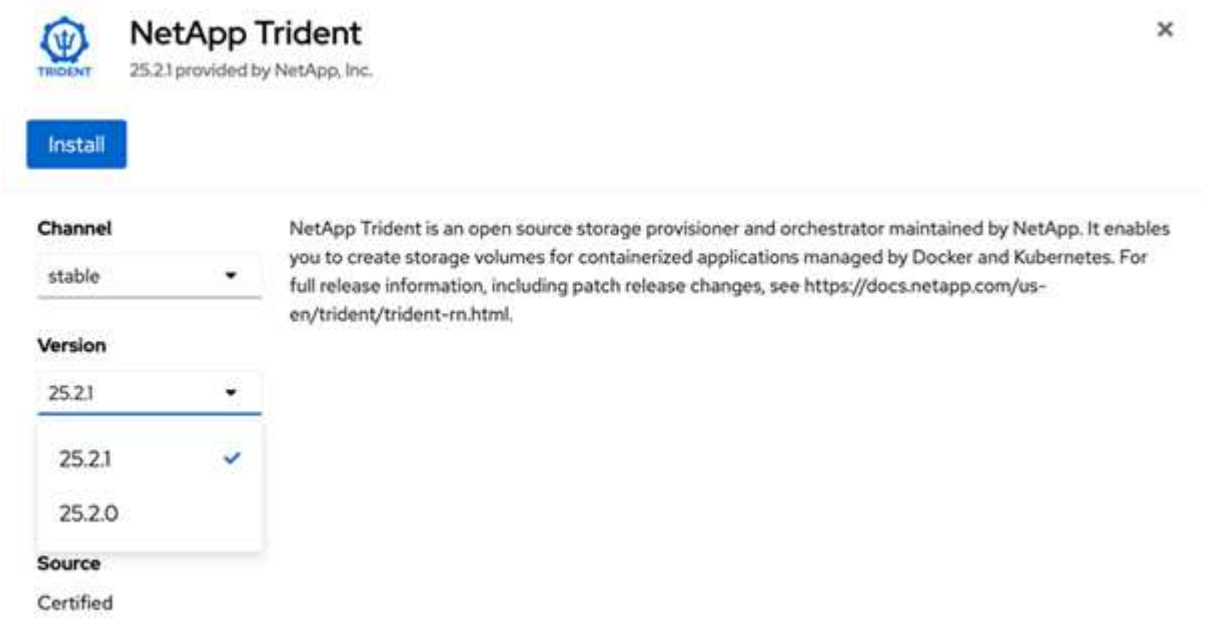
Trouvez et installez l'opérateur Trident

Étapes

1. Accédez à OpenShift OperatorHub et recherchez NetApp Trident.

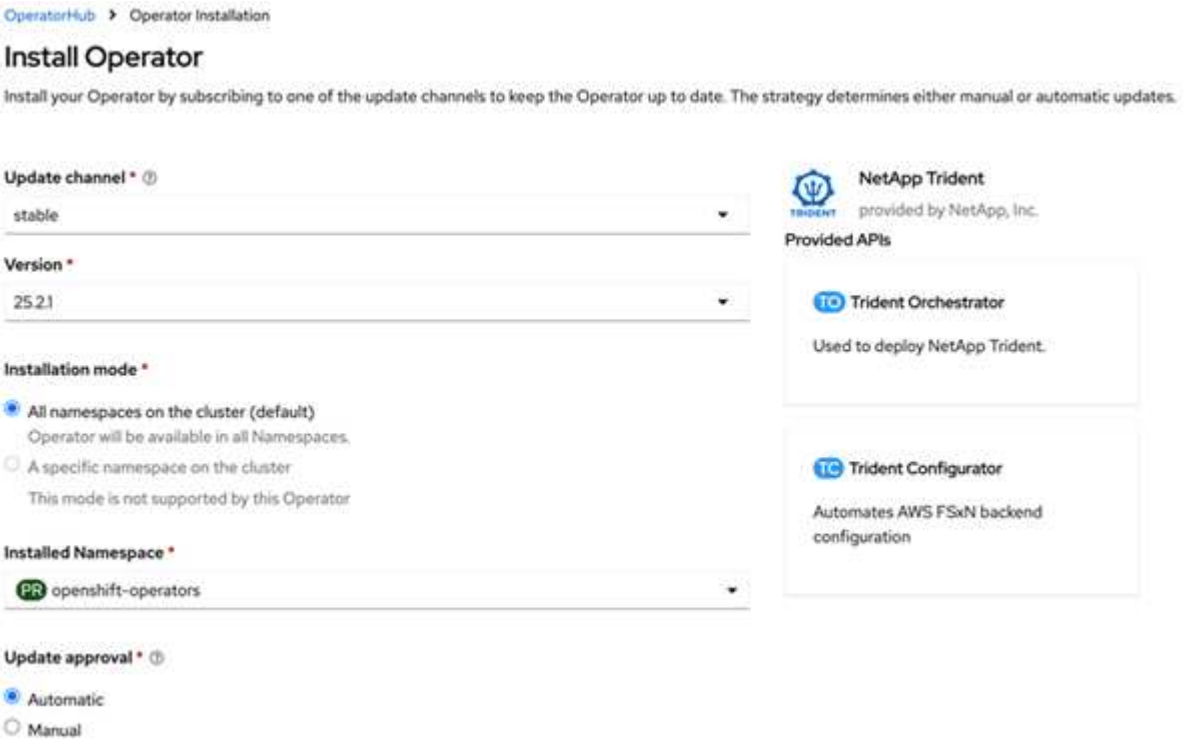


2. Cliquez sur **NetApp Trident** pour ouvrir les paramètres d'installation.
3. Sélectionnez les options requises et cliquez sur **Installer** pour ouvrir la configuration de l'opérateur.

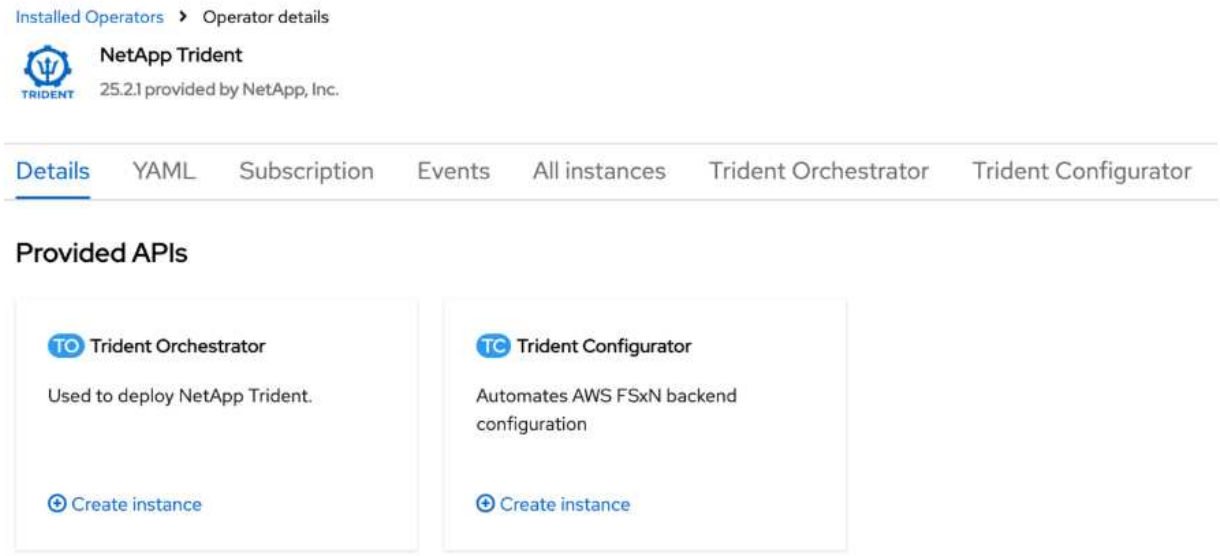


Assurez-vous de sélectionner la version la plus récente d'Operator.

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



5. Cliquez sur **View Operator** pour afficher les détails de l'Operator.



6. Cliquez sur **YAML view** 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.

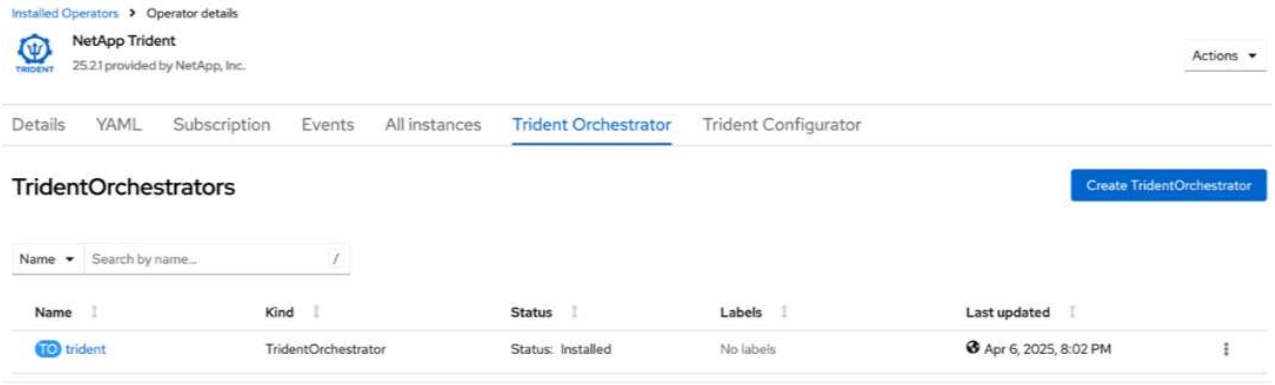
Facultatif : Activer la concurrence

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 pour configurer et activer les services iSCSI et Multipath sur tous les nœuds de travail OpenShift.
- À partir de OpenShift 4.19, la version minimale de Trident prise en charge pour cette fonctionnalité est 25.06.1.

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



Désinstaller l'opérateur Trident

Étapes

1. Sélectionnez l'opérateur Trident dans la liste des opérateurs installés.
2. Sélectionnez si vous souhaitez 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 **Uninstall**.

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

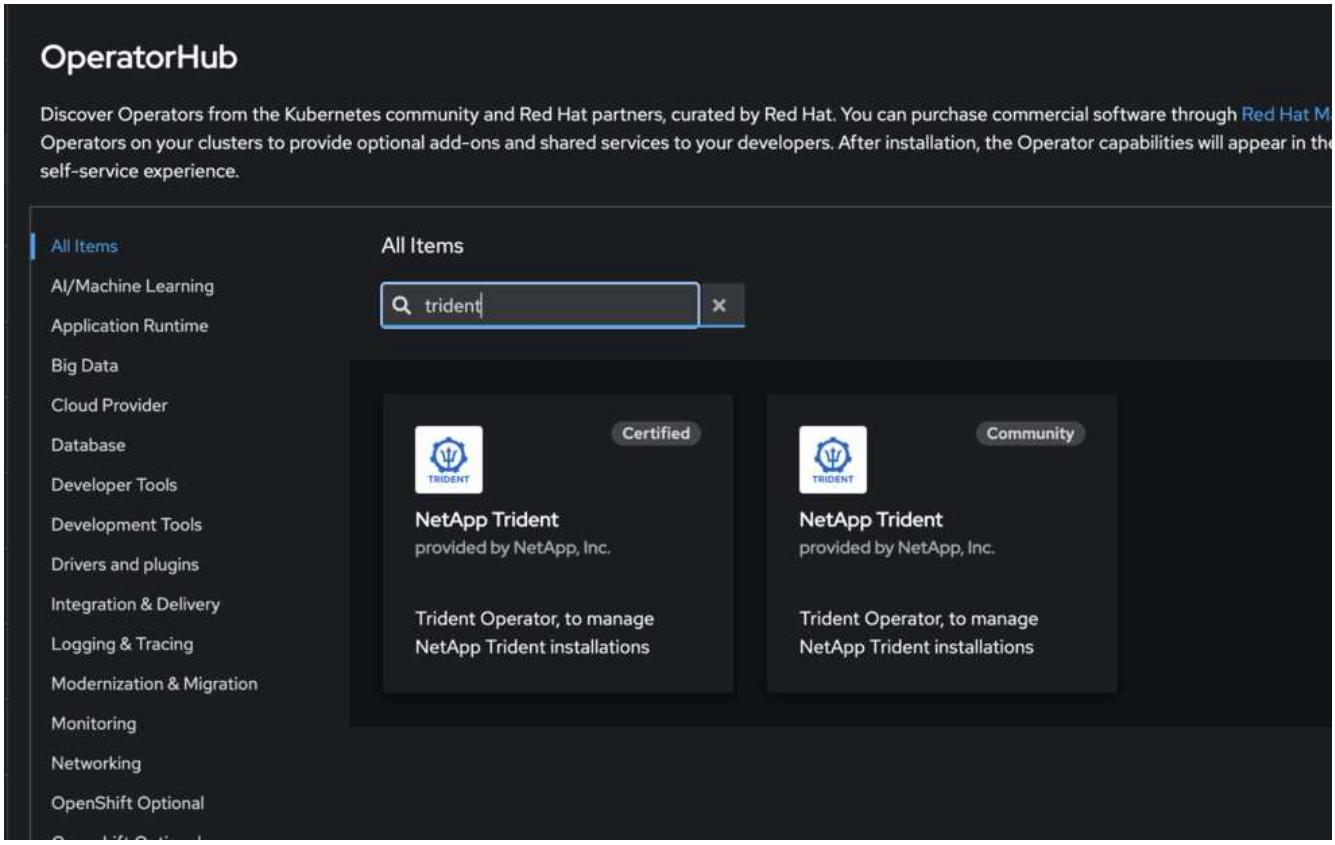
Avant de commencer l'installation, "Préparez votre environnement pour l'installation de Trident".

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.

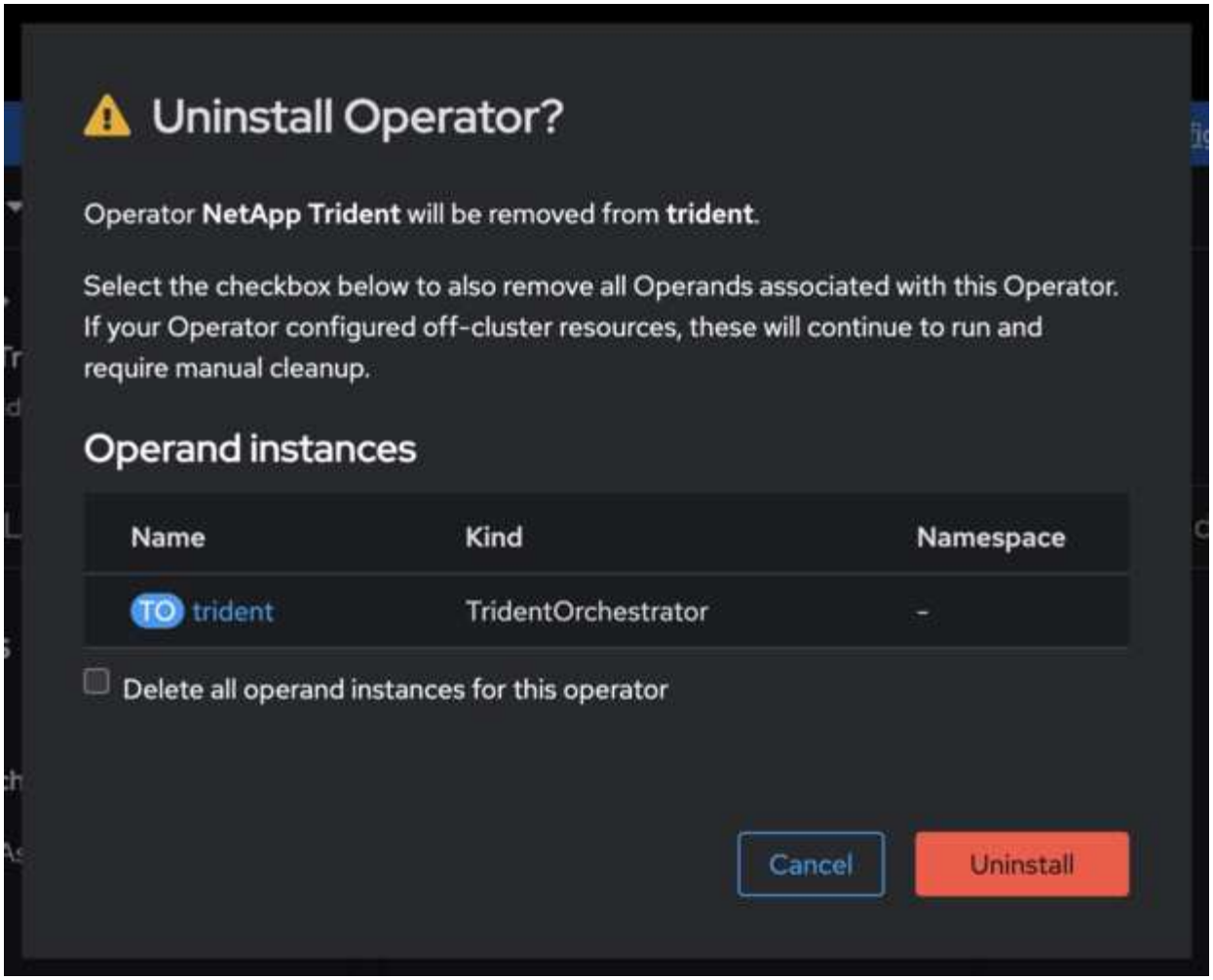
Passez de l'opérateur communautaire

Étapes

1. Utilisez la console OpenShift pour naviguer vers le OperatorHub.



2. Trouvez l’opérateur communautaire NetApp Trident.



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

3. Cliquez sur **Uninstall**.
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 :

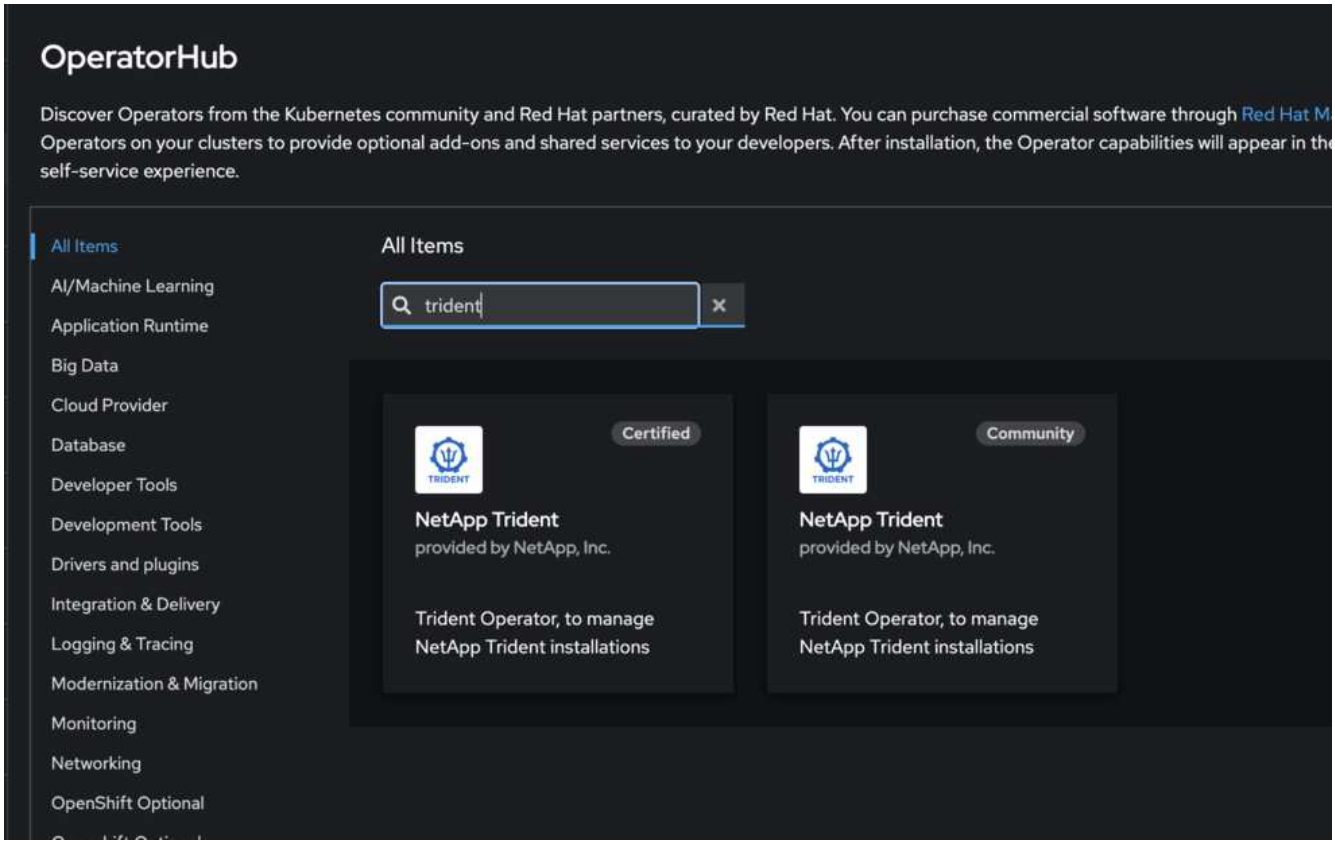
```
kubectl delete -f deploy/bundle.yaml -n trident
```

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

Installez l'opérateur certifié OpenShift

Étapes

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



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

Vérification

- Vérifiez dans la console OperatorHub pour vous assurer que le nouvel opérateur certifié a bien été installé avec succès.

Utilisez Trident

Préparez le nœud de travail

Tous les nœuds de travail du cluster Kubernetes doivent pouvoir monter les volumes que vous avez provisionnés pour vos pods. Pour préparer les nœuds de travail, vous devez installer NFS, iSCSI, NVMe/TCP ou les outils FC en fonction de votre sélection de pilote.

Choisir les bons outils

Si vous utilisez une combinaison de pilotes, vous devez installer tous les outils requis pour vos pilotes. Les versions récentes de Red Hat Enterprise Linux CoreOS (RHCOS) ont les outils installés par défaut.

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 nonvolatile memory express (NVMe) sur TCP (NVMe/TCP).

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

Outils SCSI sur FC

Consultez "Méthodes de configuration des hôtes SAN FC et FC-NVMe" pour plus d'informations sur la configuration de vos hôtes SAN FC et 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 OpenShift et KubeVirt environnements. * SCSI sur FC n'est pas pris en charge sur Docker. * L'auto-réparation iSCSI n'est pas applicable à SCSI sur FC.

Outils SMB

"Préparez-vous à provisionner des volumes SMB" si vous utilisez : `ontap-nas` pour provisionner des volumes SMB.

Découverte de service de nœud

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

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

Revoir les événements

Trident crée des événements pour le nœud afin d'identifier les services découverts. Pour consulter 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 Trident node CR. Pour afficher les services détectés, exécutez :

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

Volumes NFS

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

RHEL 8+

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

Ubuntu

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

Redémarrez vos nœuds de travail après l'installation des outils NFS pour éviter les échecs lors de l'attachement des volumes aux conteneurs.

volumes iSCSI

Trident peut établir automatiquement une session iSCSI, analyser les LUN, découvrir les périphériques multipath, les formater et les monter à un pod.

Capacités d'auto-réparation iSCSI

Pour les systèmes ONTAP, Trident exécute une auto-réparation iSCSI toutes les cinq minutes afin de :

1. **Identifier** l'état de session iSCSI souhaité et l'état de session iSCSI actuel.
2. **Comparez** l'état souhaité à l'état actuel pour identifier les réparations nécessaires. Trident détermine les priorités de réparation et le moment où il faut préempter les réparations.
3. **Effectuer les réparations** nécessaires pour ramener l'état actuel de la session iSCSI à l'état souhaité de la session iSCSI.

Les journaux d'activité d'auto-réparation se trouvent dans le `trident-main` conteneur sur le pod Daemonset respectif. Pour afficher les journaux, vous devez avoir défini `debug` sur "true" lors de l'installation de Trident.

Les capacités d'auto-réparation iSCSI de Trident peuvent aider à prévenir :

- Des sessions iSCSI obsolètes ou défaillantes qui pourraient survenir après un problème de connectivité réseau. En cas de session obsolète, Trident attend sept minutes avant de se déconnecter pour rétablir la connexion avec un portail.

  Par exemple, si les secrets CHAP sont renouvelés sur le contrôleur de stockage et que le réseau perd la connectivité, les anciens secrets CHAP (obsolètes) peuvent persister. L'auto-réparation peut reconnaître cela 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-réparation iSCSI lancera des analyses SCSI pour tous les périphériques du bus SCSI.
- Si seuls les igroups à portée backend (dépréciés depuis 23.04) sont utilisés, l'auto-réparation iSCSI lancera des analyses SCSI pour les ID LUN exacts sur le bus SCSI.
- Si une combinaison d'igroups par nœud et d'igroups à portée backend est utilisée, l'auto-réparation iSCSI lancera des analyses SCSI pour les ID LUN exacts sur le bus SCSI.

Installez les outils iSCSI

Installez les outils iSCSI en utilisant les commandes pour votre système d'exploitation.

Avant de commencer

- Chaque nœud du cluster Kubernetes doit posséder un IQN unique. **Il s'agit d'une condition préalable indispensable**.
- Si vous utilisez RHCOS version 4.5 ou ultérieure, ou une autre distribution Linux compatible RHEL, avec le `solidfire-san` pilote et Element OS 12.5 ou une 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 de travail exécutant RHEL/Red Hat Enterprise Linux CoreOS (RHCOS) avec des volumes persistants iSCSI, spécifiez la `discard` mountOption dans la StorageClass pour effectuer la récupération d'espace en ligne. Consultez "Documentation Red Hat".
- Assurez-vous d'avoir effectué la mise à jour vers la dernière version du `multipath-tools`.

RHEL 8+

1. Installez les paquets système suivants :

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

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

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

3. Définissez la numérisation sur manuel :

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

4. Activez le multipathing :

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

> Assurez-vous `/etc/multipath.conf` contient `find_multipaths no` sous `defaults`.

5. Assurez-vous 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 paquets système suivants :

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

2. Vérifiez que la version d'open-iscsi est 2.0.874-5ubuntu2.10 ou ultérieure (pour bionic) ou 2.0.874-7.1ubuntu6.1 ou ultérieure (pour focal):

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

3. Définissez la numérisation sur manuel :

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

4. Activez le multipathing :

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

5. Assurez-vous que `open-iscsi` et `multipath-tools` sont activés 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 démarrer `open-iscsi` pour que le démon iSCSI démarre. Vous pouvez également modifier le service `iscsi` pour qu'il démarre `iscsid` automatiquement.

Configurer ou désactiver l'autoréparation iSCSI

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

- **Intervalle d'auto-réparation iSCSI** : détermine la fréquence à laquelle l'auto-réparation iSCSI est invoquée (par défaut : 5 minutes). Vous pouvez le configurer pour qu'il s'exécute plus fréquemment en définissant une valeur plus petite ou moins fréquemment en définissant une valeur plus grande.

Définir l'intervalle d'auto-réparation iSCSI à 0 désactive complètement l'auto-réparation iSCSI. Nous déconseillons de désactiver l'auto-réparation iSCSI ; elle ne doit être désactivée que dans certains scénarios lorsque l'auto-réparation iSCSI ne fonctionne pas comme prévu ou à des fins de débogage.

- **Délai d'attente d'auto-réparation iSCSI** : détermine la durée pendant laquelle l'auto-réparation iSCSI attend avant de se déconnecter d'une session défaillante et de tenter de se reconnecter (par défaut : 7 minutes). Vous pouvez le configurer sur une valeur plus élevée afin que les sessions identifiées comme défaillantes doivent attendre plus longtemps avant d'être déconnectées puis qu'une tentative de reconnexion soit effectuée, ou sur une valeur plus faible pour se déconnecter et se reconnecter plus tôt.

Helm

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

L'exemple suivant configure l'intervalle d'auto-réparation iSCSI à 3 minutes et le délai d'attente d'auto-réparation à 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-réparation iSCSI, transmettez les `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 configure l'intervalle d'auto-réparation iSCSI à 3 minutes et le délai d'attente d'auto-réparation à 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 pour votre système d'exploitation.

- NVMe nécessite 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 vers une version 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
```

Vérifier l'installation

Après l'installation, vérifiez que chaque nœud du cluster Kubernetes possède un NQN unique à l'aide de la commande :

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

Trident modifie la valeur `ctrl_device_tmo` pour garantir 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 le système ONTAP.

Prérequis

Configurez les paramètres réseau et de nœud requis pour FC.

Paramètres réseau

1. Obtenez le WWPN des interfaces cibles. Consultez "network interface show" pour plus d'informations.
2. Obtenez le WWPN pour les interfaces sur l'initiateur (Host).

   Consultez les utilitaires correspondants du système d'exploitation hôte.

3. Configurez le zonage sur le commutateur FC en utilisant les WWPN du Host et du target.

   Reportez-vous à la documentation du fournisseur du commutateur concerné pour obtenir des informations.

   Pour plus de détails, veuillez consulter la documentation ONTAP suivante :

   - "Aperçu du zonage Fibre Channel et FCoE"
   - "Méthodes de configuration des hôtes SAN FC et FC-NVMe"

Installez les outils FC

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

- Lors de l'utilisation de nœuds de travail exécutant RHEL/Red Hat Enterprise Linux CoreOS (RHCOS) avec des FC PVs, spécifiez le `discard` mountOption dans le StorageClass pour effectuer une récupération d'espace en ligne. Consultez "Documentation Red Hat".

RHEL 8+

1. Installez les paquets système suivants :

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

2. Activez le multipathing :

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

> Assurez-vous `/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 paquets système suivants :

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

2. Activez le multipathing :

```
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 `/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
```

Préparez-vous à provisionner des volumes SMB

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

> Vous devez configurer les protocoles NFS et SMB/CIFS sur la SVM pour créer un `ontap-nas-economy` volume SMB pour les clusters ONTAP sur site. L'absence de configuration de l'un ou l'autre de ces protocoles entraînera 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.

- Un cluster Kubernetes avec un nœud contrôleur Linux et au moins un nœud de travail Windows exécutant Windows Server 2022. Trident prend uniquement en charge les volumes SMB montés sur des pods exécutés sur des nœuds Windows.
- Au moins un secret Trident contenant vos identifiants Active Directory. Pour générer le 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`, reportez-vous à "GitHub : CSI Proxy" ou "GitHub: CSI Proxy pour Windows" pour les nœuds Kubernetes exécutés sous Windows.

Étapes

1. Pour ONTAP sur site, vous pouvez éventuellement créer un partage SMB ou Trident peut 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 : soit à l'aide du "Microsoft Management Console" Shared Folders snap-in, soit 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 commande ONTAP :

   a. Si nécessaire, créez la structure du chemin d'accès du répertoire pour le partage.

      La `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 à la SVM spécifiée :

```
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 "Créer un partage SMB" pour plus de détails.

2. Lors de la création du backend, vous devez configurer les éléments suivants pour spécifier les volumes SMB. Pour toutes les options de configuration du backend FSx pour ONTAP, reportez-vous à "Options de configuration et exemples pour FSx for 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 pour permettre à Trident de créer le partage SMB ; ou vous pouvez laisser le paramètre vide pour empêcher l'accès partagé aux volumes. Ce paramètre est facultatif pour ONTAP sur site. Ce paramètre est obligatoire pour Amazon FSx for ONTAP backends et ne peut pas être vide.	`smb-share`
`nasType`	**Doit être défini sur `smb`.** Si nul, 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` or `mixed` pour les volumes SMB
`unixPermissions`	Mode pour les nouveaux volumes. **Doit rester vide pour les volumes SMB.**	""

Configurer et gérer les backends

Configurer les backends

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

Trident propose automatiquement des pools de stockage provenant de backends correspondant aux exigences définies par une classe de stockage. Découvrez comment configurer le backend pour votre système de stockage.

- "Configurer un backend Azure NetApp Files"
- "Configurer un backend Google Cloud NetApp Volumes"
- "Configurer un backend NetApp HCI ou SolidFire"
- "Configurez un backend avec les pilotes NAS ONTAP ou Cloud Volumes ONTAP"
- "Configurez un backend avec les pilotes SAN ONTAP ou Cloud Volumes ONTAP"

- "Utilisez Trident avec Amazon FSx for NetApp ONTAP"

Azure NetApp Files

Configurer un backend 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).

Pilote	Protocole	volumeMode	Modes d'accès pris en charge	Systèmes de fichiers pris en charge
`azure-netapp-files`	NFS SMB	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 uniquement en charge les volumes SMB montés sur des pods exécutés sur des nœuds Windows.
- 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 :

- Un 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"
```

Helm

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 :

- Un cluster Kubernetes déployé à l'aide d'AKS
- Identité de la charge de travail et oidc-issuer configurés sur le cluster Kubernetes AKS

- 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
```

Helm

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-vous à configurer un backend Azure NetApp Files

Avant de pouvoir configurer votre backend Azure NetApp Files, vous devez vous assurer que les conditions suivantes sont remplies.

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 : Configurer Azure NetApp Files et créer 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 facultatifs lors de l'utilisation d'identités gérées sur un cluster AKS.
- `tenantID`, `clientID`, et `clientSecret` sont facultatifs lors de l'utilisation d'une identité 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. Consultez "Microsoft : Créer un pool de capacité pour Azure NetApp Files".
- Un sous-réseau délégué à Azure NetApp Files. Consultez "Microsoft : Déléguer un sous-réseau à Azure NetApp Files".
- `subscriptionID` à partir d'un abonnement Azure avec Azure NetApp Files activé.
- `tenantID`, `clientID`, et `clientSecret` à partir d'un "Inscription de l'application" dans Azure Active Directory disposant des autorisations suffisantes pour le service Azure NetApp Files. L'inscription de l'application doit utiliser l'une des méthodes suivantes :
  - Le rôle de propriétaire ou de contributeur "prédéfini par Azure".
  - Un "Rôle Contributor personnalisé" au niveau de l'abonnement (`assignableScopes`) avec les autorisations suivantes, limitées uniquement à ce dont Trident a besoin. 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": []
        }
      ]
    }
}
```

- L'Azure `location` qui contient au moins un "sous-réseau délégué". À partir de Trident 22.01, le paramètre `location` est un champ obligatoire au niveau supérieur du fichier de configuration du backend. Les valeurs d'emplacement spécifiées dans les pools virtuels sont ignorées.
- Pour utiliser `Cloud Identity`, obtenez l' `client ID` à 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 de :

- Active Directory configuré et connecté à Azure NetApp Files. Consultez "Microsoft : Créer et gérer des connexions Active Directory pour Azure NetApp Files".
- Un cluster Kubernetes avec un nœud contrôleur Linux et au moins un nœud de travail Windows exécutant Windows Server 2022. Trident prend uniquement en charge les volumes SMB montés sur des pods exécutés sur des nœuds Windows.
- Au moins un secret Trident contenant vos identifiants Active Directory afin qu'Azure NetApp Files puisse s'authentifier auprès d'Active Directory. Pour générer le 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`, reportez-vous à "GitHub : CSI Proxy" ou "GitHub: CSI Proxy pour Windows" pour les nœuds Kubernetes exécutés sous Windows.

Options et exemples de configuration du backend Azure NetApp Files

Découvrez les options de configuration NFS et SMB pour Azure NetApp Files et consultez des exemples de configuration.

Options de configuration du backend

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 backends Azure NetApp Files offrent ces options de configuration.

Paramètre	Description	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	Driver name + "_" + caractères aléatoires
`subscriptionID`	L'ID d'abonnement de votre abonnement Azure Facultatif lorsque les identités gérées sont activées sur un cluster AKS.	
`tenantID`	L'identifiant du locataire issu d'un enregistrement d'application. Facultatif lorsque des identités gérées ou une identité cloud sont utilisées sur un cluster AKS.	
`clientID`	L'identifiant client issu d'un enregistrement d'application, facultatif lorsque des identités gérées ou une identité cloud sont utilisées sur un cluster AKS.	
`clientSecret`	Le secret client issu d'un App Registration est facultatif lorsque des identités gérées ou une cloud identity sont utilisées sur un cluster AKS.	
`serviceLevel`	L'un de `Standard`, `Premium` ou `Ultra`	"" (aléatoire)
`location`	Nom de l'emplacement Azure où les nouveaux volumes seront créés Facultatif lorsque les identités gérées sont activées sur un cluster AKS.	
`resourceGroups`	Liste des groupes de ressources pour filtrer les ressources découvertes	"[]" (aucun filtre)

Paramètre	Description	Défaut
`netappAccounts`	Liste des comptes NetApp pour le filtrage des ressources découvertes	"[]" (aucun filtre)
`capacityPools`	Liste des pools de capacité pour le filtrage des ressources découvertes	"[]" (no filter, 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`	Échec de l'approvisionnement si la taille du volume demandée est supérieure à cette valeur	"" (non appliqué par défaut)
`debugTraceFlags`	Options de débogage à utiliser lors du dépannage. Exemple, `\{"api": false, "method": true, "discovery": true}`. N'utilisez cette option que si vous effectuez un dépannage et avez besoin d'un journal détaillé.	null
`nasType`	Configurez la création de volumes NFS ou SMB. Les options sont `nfs`, `smb` ou null. La valeur null correspond par défaut à des volumes NFS.	`nfs`

Paramètre	Description	Défaut
`supportedTopologies`	Représente une liste de régions et de zones prises en charge par ce backend. Pour plus d'informations, consultez "Utiliser la topologie CSI".	
`qosType`	Représente le type de QoS : Auto ou Manual.	Automatique
`maxThroughput`	Définit le débit maximal autorisé en Mio/s. Pris en charge uniquement pour les pools de capacité QoS manuels.	`4 MiB/sec`

Pour plus d'informations sur les fonctionnalités réseau, veuillez consulter "Configurer les fonctionnalités réseau pour 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 vNet 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 resourceGroups 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	<resource group>
Compte NetApp	<resource group>/<netapp account>
Pool de capacité	<resource group>/<netapp account>/<capacity pool>
Réseau virtuel	<resource group>/<virtual network>
Sous-réseau	<resource group>/<virtual network>/<subnet>

Provisionnement de volumes

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	Défaut
`exportRule`	Règles d'exportation pour les nouveaux volumes. `exportRule` doit être une liste séparée par des virgules de toute 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`	La taille par défaut des nouveaux volumes	"100G"
`unixPermissions`	Les permissions Unix des nouveaux volumes (4 chiffres octaux). Ignoré pour les volumes SMB.	"" (fonctionnalité en avant-première, nécessite une inscription sur la liste blanche 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 débutez avec Azure NetApp Files et que vous faites des essais, mais en pratique vous souhaiterez définir une portée supplémentaire pour les 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 managées pour AKS

Cette configuration backend omet `subscriptionID`, `tenantID`, `clientID` et `clientSecret`, qui sont optionnels 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 backend omet `tenantID`, `clientID` et `clientSecret`, qui sont optionnels lors de l'utilisation d'une identité cloud.

```
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 spécifique du niveau de service 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 des pools de capacité QoS manuels

Cette configuration backend place les volumes dans l'emplacement `eastus` Azure avec des 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 backend réduit encore la portée du placement des volumes à un seul sous-réseau et modifie également certains paramètres par défaut de 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 du 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 `StorageClass` définitions suivantes se rapportent aux pools de stockage ci-dessus.

Exemple de définitions utilisant `parameter.selector` field

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, 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 à 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.

Aperçu

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-vous à configurer

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"
```

Helm

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)

Pilote	Protocole	volumeMode	Modes d'accès pris en charge	Systèmes de fichiers pris en charge
`google-cloud-netapp-volumes`	NFS SMB	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.

Aperçu

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
- Instantané
- 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-vous à configurer un backend Google Cloud NetApp Volumes

Avant de pouvoir configurer votre backend Google Cloud NetApp Volumes, vous devez vous assurer que les conditions suivantes sont remplies.

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".

Assurez-vous de disposer des éléments suivants avant de configurer Google Cloud NetApp Volumes backend :

- Un compte Google Cloud configuré avec le service Google Cloud NetApp Volumes. Consultez "Google Cloud NetApp Volumes".
- Numéro de projet de votre compte Google Cloud. Consultez "Identification des projets".
- Un compte de service Google Cloud avec le rôle NetApp Volumes Admin (`roles/netapp.admin`. Consultez "Rôles et autorisations d'Identity and Access Management".
- Fichier de clé API pour votre compte GCNV. Consultez "Créer une clé de compte de service"
- Un pool de stockage. Consultez "Aperçu des pools de stockage".

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

Options et exemples de configuration du backend Google Cloud NetApp Volumes

Découvrez les options de configuration backend pour Google Cloud NetApp Volumes et consultez des exemples de configuration.

Options de configuration du backend

Chaque backend provisionne des volumes dans une seule région Google Cloud. Pour créer des volumes dans d'autres régions, vous pouvez définir des backends supplémentaires.

Paramètre	Description	Défaut
`version`		Toujours 1
`storageDriverName`	Nom du pilote de stockage	La valeur de `storageDriverName` doit être spécifiée comme « google-cloud-netapp-volumes ».
`backendName`	(Facultatif) Nom personnalisé du stockage backend	Nom du driver + "_" + partie de la clé API
`storagePools`	Paramètre optionnel utilisé pour spécifier les pools de stockage pour la création de volumes.	
`projectNumber`	Numéro de projet du compte Google Cloud. La valeur se trouve sur la page d'accueil du portail Google Cloud.	
`location`	L'emplacement Google Cloud où Trident crée les volumes GCNV. Lors de la création de clusters Kubernetes interrégionaux, les volumes créés dans un `location` peuvent être utilisés dans des charges de travail planifiées sur des nœuds répartis dans plusieurs régions Google Cloud. Le trafic interrégional engendre des coûts supplémentaires.	
`apiKey`	Clé API pour le compte de service Google Cloud avec le `netapp.admin` rôle. Elle inclut le contenu au format JSON du fichier de clé privée d'un compte de service Google Cloud (copié tel quel dans le fichier de configuration du backend). 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`	L'approvisionnement échoue si la taille du volume demandée est supérieure à cette valeur.	"" (non appliqué par défaut)
`serviceLevel`	Le 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`	Options de débogage à utiliser lors du dépannage. Exemple, `{"api":false, "method":true}`. N'utilisez cette option que si vous effectuez un dépannage et avez besoin d'un journal détaillé.	null

Paramètre	Description	Défaut
`nasType`	Configurez la création de volumes NFS ou SMB. Les options sont `nfs`, `smb` ou null. La valeur null correspond par défaut à des volumes NFS.	`nfs`
`supportedTopologies`	Représente une liste de régions et de zones prises en charge par ce backend. Pour plus d'informations, consultez "Utiliser la topologie CSI". Par exemple : `supportedTopologies: - topology.kubernetes.io/region: asia-east1 topology.kubernetes.io/zone: asia-east1-a`	

Options de provisionnement de volume

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

Paramètre	Description	Défaut
`exportRule`	Les 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 du volume réservé aux instantanés	"" (accepter la valeur par défaut de 0)
`unixPermissions`	Les permissions Unix des nouveaux volumes (4 chiffres octaux).	""

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 pools de stockage délégués à Google Cloud NetApp Volumes à l'emplacement configuré, et place les nouveaux volumes sur l'un de ces pools de façon aléatoire. 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 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 le 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 du pool virtuel

Cette configuration backend définit plusieurs pools virtuels dans un seul fichier. Les pools virtuels sont définis dans la `storage` section. Ils sont utiles lorsque vous disposez de plusieurs pools de stockage prenant en charge différents niveaux de service et que vous souhaitez créer des classes de stockage dans Kubernetes pour les représenter. Les étiquettes des pools virtuels sont utilisées pour différencier les pools. Par exemple, dans l'exemple ci-dessous, `performance` label et `serviceLevel` 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 pour 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, consultez "Pools virtuels".

```
---
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

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

Configuration des topologies prises en charge

Trident facilite le provisionnement des volumes pour les 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: 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 ensuite ?

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

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

Pour vérifier que le backend a bien été 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 backend échoue, cela signifie qu'il y a un problème avec la configuration du backend. Vous pouvez décrire le backend à l'aide de la commande `kubectl get tridentbackendconfig <backend-name>` ou consulter les journaux pour déterminer la cause en exécutant la commande suivante :

```
tridentctl logs
```

Après avoir identifié et corrigé le problème lié au fichier de configuration, vous pouvez supprimer le backend et exécuter à nouveau la commande de création.

Définitions des classes de stockage

L'exemple suivant est une `StorageClass`définition de base qui fait référence au backend 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 champ `parameter.selector` :

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: 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 à "Créer une classe de stockage".

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. Tout utilisateur/mot de passe Active Directory, avec ou sans autorisations, peut être utilisé pour le secret de l'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 prenant en charge les volumes SMB. `nasType: nfs ou `nasType: null`filtres pour les pools NFS.

Exemple de définition du PVC

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

Pour vérifier si le PVC est lié, 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.

Aperçu

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 un backend NetApp HCI ou SolidFire

Apprenez à créer et à utiliser un backend Element avec votre installation Trident.

Détails du pilote Element

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

Le `solidfire-san` pilote de stockage prend en charge les modes de volume *file* et *block*. Pour le

`Filesystem` volumeMode, Trident crée un volume et crée un système de fichiers. Le type de système de fichiers est spécifié par le StorageClass.

Pilote	Protocole	VolumeMode	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 backend Element.

- Un système de stockage pris en charge qui exécute le logiciel Element.
- Identifiants d'un administrateur de cluster NetApp HCI/SolidFire ou d'un utilisateur locataire pouvant gérer les volumes.
- Tous vos nœuds de travail Kubernetes doivent disposer des outils iSCSI appropriés. Consultez "Informations de préparation du nœud de travail".

Options de configuration du backend

Consultez le tableau suivant pour les options de configuration du backend :

Paramètre	Description	Défaut
`version`		Toujours 1
`storageDriverName`	Nom du pilote de stockage	Toujours "solidfire-san"
`backendName`	Nom personnalisé ou le stockage backend	"solidfire_" + adresse IP de stockage (iSCSI)
`Endpoint`	MVIP pour le cluster SolidFire avec les identifiants du locataire	
`SVIP`	Adresse IP et port de stockage (iSCSI)	
`labels`	Ensemble d'étiquettes au format JSON arbitraires à appliquer aux volumes.	""
`TenantName`	Nom du tenant à utiliser (créé s'il n'est pas trouvé)	
`InitiatorIFace`	Limiter le trafic iSCSI à une interface d'hôte spécifique	"default"
`UseCHAP`	Utilisez CHAP pour authentifier iSCSI. Trident utilise CHAP.	true

Paramètre	Description	Défaut
`AccessGroups`	Liste des ID de groupes d'accès à utiliser	Recherche l'ID d'un groupe d'accès nommé "trident"
`Types`	Spécifications QoS	
`limitVolumeSize`	Échec de l'approvisionnement si la taille du volume demandée est supérieure à cette valeur	"" (non appliqué par défaut)
`debugTraceFlags`	Options de débogage à utiliser lors du dépannage. Exemple, {"api":false, "method":true}	null

AVERTISSEMENT	N'utilisez pas `debugTraceFlags` sauf si vous effectuez un dépannage et avez besoin d'un vidage détaillé du journal.

Exemple 1 : Configuration du backend pour `solidfire-san` driver avec trois types de volumes

Cet exemple illustre un fichier backend utilisant l'authentification CHAP et modélisant trois types de volumes avec des garanties QoS spécifiques. Vous définirez ensuite probablement des classes de stockage pour utiliser chacune d'elles à l'aide du `IOPS` storage class parameter.

```
---
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 backend et de la classe de stockage pour `solidfire-san` driver avec pools virtuels

Cet exemple montre le fichier de définition du backend configuré avec des pools virtuels ainsi que StorageClasses qui y font référence.

Trident copie les étiquettes présentes sur un pool de stockage vers le LUN de stockage principal lors du provisionnement. Pour plus de simplicité, les administrateurs de stockage peuvent définir des étiquettes par pool virtuel et regrouper les volumes par étiquette.

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

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

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

Les définitions StorageClass suivantes se rapportent aux pools virtuels mentionnés ci-dessus. En utilisant le

champ `parameters.selector`, chaque StorageClass indique le ou les pools virtuels pouvant héberger un volume. Le volume possédera les aspects définis dans le pool virtuel choisi.

Le premier StorageClass (`solidfire-gold-four`) correspond au premier pool virtuel. Il s'agit du seul pool offrant des performances gold avec un `Volume Type QoS` de Gold. Le dernier StorageClass (`solidfire-silver`) fait référence à tout pool de stockage offrant des performances silver. Trident déterminera quel pool virtuel est sélectionné et s'assurera que l'exigence de stockage est respectée.

```
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
```

Pour plus d'informations

- "Groupes d'accès au volume"

Pilotes ONTAP SAN

Présentation du pilote ONTAP SAN

Découvrez comment configurer un backend ONTAP avec les pilotes SAN ONTAP et Cloud Volumes ONTAP.

Détails du pilote ONTAP SAN

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

Pilote	Protocole	volumeMode	Modes d'accès pris en charge	Systèmes de fichiers pris en charge
`ontap-san`	iSCSI SCSI sur FC	Bloc	RWO, ROX, RWX, RWOP	Aucun système de fichiers ; périphérique bloc brut
`ontap-san`	iSCSI SCSI sur FC	Système de fichiers	RWO, RWOP ROX et RWX ne sont pas disponibles en mode volume du système de fichiers.	`xfs`, `ext3`, `ext4`
`ontap-san`	NVMe/TCP Consultez Considérations supplémentaires pour NVMe/TCP.	Bloc	RWO, ROX, RWX, RWOP	Aucun système de fichiers ; périphérique bloc brut

Pilote	Protocole	volumeMode	Modes d'accès pris en charge	Systèmes de fichiers pris en charge
`ontap-san`	NVMe/TCP Consultez Considérations supplémentaires pour NVMe/TCP.	Système de fichiers	RWO, RWOP ROX et RWX ne sont pas disponibles en mode volume du système de fichiers.	`xfs`, `ext3`, `ext4`
`ontap-san-economy`	iSCSI	Bloc	RWO, ROX, RWX, RWOP	Aucun système de fichiers ; périphérique bloc brut
`ontap-san-economy`	iSCSI	Système de fichiers	RWO, RWOP ROX et RWX ne sont pas disponibles en mode volume du système de fichiers.	`xfs`, `ext3`, `ext4`

AVERTISSEMENT

- Utilisez `ontap-san-economy` uniquement si le nombre d'utilisations du volume persistant doit être supérieur à "limites de volume ONTAP prises en charge".
- Utilisez `ontap-nas-economy` uniquement si le nombre d'utilisations de volumes persistants est censé être supérieur à "limites de volume ONTAP prises en charge" et que le pilote `ontap-san-economy` ne peut pas être utilisé.
- N'utilisez pas `ontap-nas-economy` si vous prévoyez un besoin de protection des données, de reprise après sinistre ou de mobilité.
- NetApp ne recommande pas d'utiliser la croissance automatique FlexVol® dans tous les pilotes ONTAP, sauf ontap-san. En guise de solution de contournement, Trident prend en charge l'utilisation de la réserve de snapshots et adapte les volumes FlexVol® en conséquence.

Autorisations de l'utilisateur

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

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

Bien qu'il soit possible de créer un rôle plus restrictif au sein d'ONTAP qu'un pilote Trident peut utiliser, nous ne le recommandons pas. La plupart des nouvelles versions de Trident feront appel à des API supplémentaires dont il faudrait tenir compte, ce qui rend les mises à niveau difficiles et sujettes aux erreurs.

Considérations supplémentaires pour NVMe/TCP

Trident prend en charge le protocole non-volatile memory express (NVMe) à l'aide du `ontap-san` driver, notamment :

- IPv6
- Instantanés 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
- Multipathing natif NVMe
- Arrêt progressif ou brutal des nœuds K8s (24.06)

Trident ne prend pas en charge :

- DH-HMAC-CHAP pris en charge nativement par NVMe
- Multipathing du device mapper (DM)
- chiffrement LUKS

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

Préparez-vous à configurer le backend avec les pilotes SAN ONTAP

Comprenez les exigences et les options d'authentification pour configurer un backend ONTAP avec des pilotes SAN ONTAP.

Exigences

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

REMARQUE	"Systèmes ASA r2" diffèrent des autres systèmes ONTAP (ASA, AFF et FAS) par l'implémentation de leur couche de stockage. Dans les systèmes ASA r2, des zones de disponibilité de stockage sont utilisées au lieu des agrégats. Consultez l'article de la "ce" Knowledge Base pour savoir comment affecter 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 une `san-dev` classe qui utilise le `ontap-`

`san` pilote et une `san-default` classe qui utilise le `ontap-san-economy` pilote.

Tous vos nœuds de travail Kubernetes doivent disposer des outils iSCSI appropriés. Consultez "Préparez le nœud de travail" pour plus de détails.

Authentifier le backend ONTAP

Trident propose deux modes d'authentification pour un backend ONTAP.

- Authentification par identifiants : le nom d'utilisateur et le mot de passe d'un utilisateur ONTAP disposant des autorisations requises. Il est recommandé d'utiliser un rôle de connexion de sécurité prédéfini, tel que `admin` ou `vsadmin` pour garantir une compatibilité maximale avec les versions d'ONTAP.
- Authentification par certificat : Trident peut également communiquer avec un cluster ONTAP à l'aide d'un certificat installé sur le backend. Dans ce cas, la définition du backend doit contenir les valeurs encodées en Base64 du certificat client, de la clé et, si utilisé (recommandé), du certificat de l'autorité de certification de confiance.

Vous pouvez mettre à jour les backends existants pour basculer entre les méthodes basées sur les identifiants et celles basées sur les certificats. Cependant, une seule méthode d'authentification est prise en charge à la fois. Pour passer à une autre méthode d'authentification, vous devez supprimer la méthode existante de la configuration du backend.

AVERTISSEMENT	Si vous tentez de fournir **à la fois des informations d'identification et des certificats**, la création du backend échouera 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 identifiants

Trident requiert les identifiants d'un administrateur au niveau SVM ou au niveau cluster pour communiquer avec le backend ONTAP. Il est recommandé d'utiliser des rôles standard prédéfinis tels que `admin` ou `vsadmin`. Cela garantit la compatibilité avec les futures versions d'ONTAP qui pourraient exposer des API de fonctionnalités à utiliser par les futures versions de Trident. Un rôle de connexion de sécurité personnalisé peut être créé et utilisé avec Trident, mais cela n'est pas recommandé.

Voici un exemple de définition de backend :

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"
}
```

Il est important de noter que la définition du backend est le seul endroit où les identifiants sont stockés en clair. Après la création du backend, les noms d'utilisateur et mots de passe sont encodés avec Base64 et stockés comme secrets Kubernetes. La création ou la mise à jour d'un backend est la seule étape qui nécessite la connaissance des identifiants. En tant que telle, il s'agit d'une opération réservée à l'administrateur Kubernetes/stockage.

Activer l'authentification basée sur les certificats

Les nouveaux et les anciens backends peuvent utiliser un certificat et communiquer avec le backend ONTAP. Trois paramètres sont requis dans la définition du backend.

- clientCertificate: Valeur codée en Base64 du certificat client.
- clientPrivateKey : valeur encodée en Base64 de la clé privée associée.
- trustedCACertificate : valeur encodée en Base64 du certificat CA de confiance. Si vous utilisez un CA de confiance, ce paramètre doit être fourni. Cela peut être ignoré si aucun CA de confiance n'est utilisé.

Un workflow typique implique 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 à authentifier.

```
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. Cette opération peut déjà être effectuée par l'administrateur de stockage. Ignorez si aucune autorité de certification de confiance 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 (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. Confirmez que le rôle de connexion de sécurité ONTAP prend en charge `cert` la méthode 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 du certificat généré. Remplacez <ONTAP Management LIF> et <vserver name> par l'adresse IP de la LIF de gestion et le nom SVM.

```
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 d'autorité de certification 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 backend en utilisant les valeurs obtenues à 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 |
+------------+----------------+--------------------------------------
+--------+---------+
```

Mettez à jour les méthodes d'authentification ou faites pivoter les identifiants

Vous pouvez mettre à jour un backend existant pour utiliser une méthode d'authentification différente ou pour faire pivoter ses identifiants. Cela fonctionne dans les deux sens : les backends qui utilisent un nom d'utilisateur/mot de passe peuvent être mis à jour pour utiliser des certificats ; les backends qui utilisent des certificats peuvent être mis à jour pour utiliser un nom d'utilisateur/mot de passe. Pour ce faire, vous devez supprimer la méthode d'authentification existante et ajouter la nouvelle méthode d'authentification. Ensuite, utilisez le fichier backend.json mis à jour contenant les paramètres requis pour 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 du renouvellement 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 backend. Lors du renouvellement des certificats, plusieurs certificats peuvent être associés à l'utilisateur. Le backend est ensuite mis à jour pour utiliser le nouveau certificat, après quoi l'ancien certificat peut être supprimé du cluster ONTAP.

La mise à jour d'un backend n'interrompt pas l'accès aux volumes déjà créés et n'affecte pas les connexions de volumes établies ultérieurement. Une mise à jour réussie du backend indique que Trident peut communiquer avec le backend ONTAP et gérer les futures opérations sur les volumes.

Créer un rôle ONTAP personnalisé pour Trident

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

Reportez-vous à "Générateur de rôles personnalisés Trident" pour plus d'informations sur la création de rôles personnalisés Trident.

Utilisation de l'interface de ligne de commande ONTAP

1. Créez un nouveau 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. Associez le rôle à l'utilisateur :

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

Utilisation de System Manager

Effectuez les étapes suivantes dans ONTAP System Manager :

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

      (Ou) Pour créer un rôle personnalisé au niveau de la SVM, sélectionnez **Storage > Storage VMs > `required SVM`> Paramètres > Utilisateurs et rôles**.

   b. Sélectionnez l'icône flèche (→) à côté de **Users and Roles**.
   c. Sélectionnez **+Add** sous **Roles**.
   d. Définissez les règles du rôle et cliquez sur **Save**.
2. **Associez le rôle à l'utilisateur Trident** : + Effectuez les étapes suivantes sur la page **Utilisateurs et rôles** :
   a. Sélectionnez l'icône Ajouter **+** sous **Users**.
   b. Sélectionnez le nom d'utilisateur requis, puis sélectionnez un rôle dans le menu déroulant pour **Rôle**.
   c. Cliquez sur **Enregistrer**.

Reportez-vous aux pages suivantes pour plus d'informations :

- "Rôles personnalisés pour l'administration d'ONTAP" ou "Définir des rôles personnalisés"
- "Travailler avec les rôles et les utilisateurs"

Authentifiez les connexions avec CHAP bidirectionnel

Trident peut authentifier les sessions iSCSI avec CHAP bidirectionnel pour les `ontap-san` et `ontap-san-economy` pilotes. Cela nécessite l'activation de l'option `useCHAP` dans la définition de votre backend. Lorsqu'elle est définie sur `true`, Trident configure la sécurité par défaut de l'initiateur du SVM sur CHAP bidirectionnel et définit le nom d'utilisateur et les secrets à partir du fichier backend. NetApp recommande

d'utiliser CHAP bidirectionnel pour authentifier les 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. Il est défini sur faux par défaut. Après l'avoir défini sur vrai, vous ne pouvez plus le définir sur faux.

En plus de `useCHAP=true`, les `chapInitiatorSecret`, `chapTargetInitiatorSecret`, `chapTargetUsername` et `chapUsername`, les champs doivent être inclus dans la définition du backend. Les secrets peuvent être modifiés après la création d'un backend en exécutant `tridentctl update`.

Comment ça marche

En définissant `useCHAP` sur true, l'administrateur de stockage demande à Trident de configurer CHAP sur le système de stockage. Cela inclut les éléments suivants :

- Configuration de CHAP sur la SVM :
  - Si le type de sécurité par défaut de l'initiateur SVM est « aucun » (défini par défaut) **et** qu'il n'y a pas de LUN préexistants déjà présents dans le volume, Trident définira le type de sécurité par défaut à `CHAP` et procédera à la configuration du nom d'utilisateur et des secrets de l'initiateur CHAP et de la cible.
  - Si la SVM contient des LUN, Trident n'activera pas CHAP sur la SVM. Cela garantit que l'accès aux LUN déjà présentes sur la SVM n'est pas restreint.
- Configuration du nom d'utilisateur et des secrets de l'initiateur et de la cible CHAP ; ces options doivent être spécifiées dans la configuration du backend (comme indiqué ci-dessus).

Après la création du backend, Trident crée une CRD correspondante `tridentbackend` et stocke les secrets CHAP ainsi que les noms d'utilisateur en tant que secrets Kubernetes. Tous les PV créés par Trident sur ce backend seront montés et attachés via CHAP.

Renouvelez les identifiants et mettez à jour les backends

Vous pouvez mettre à jour les informations d'identification CHAP en modifiant les paramètres CHAP dans le `backend.json` fichier. Cela nécessitera la mise à jour des secrets CHAP et l'utilisation de la `tridentctl update` commande pour appliquer 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 à l'aide de l'ONTAP CLI ou d'ONTAP System Manager, car Trident ne pourra pas prendre en compte 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 restent inchangées ; elles demeureront actives si les identifiants sont mis à jour par Trident sur le SVM. Les nouvelles connexions utilisent les identifiants mis à jour et les connexions existantes demeurent actives. La déconnexion puis la reconnexion d'anciens PV entraînera leur utilisation des identifiants mis à jour.

Options et exemples de configuration 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	Seul le `ontap-san`pilote (avec les protocoles iSCSI, NVMe/TCP et FC) est pris en charge pour les systèmes ASA r2.

Options de configuration du backend

Consultez le tableau suivant pour les options de configuration du backend :

Paramètre	Description	Défaut
`version`		Toujours 1
`storageDriverName`	Nom du pilote de stockage	`ontap-san` ou `ontap-san-economy`
`backendName`	Nom personnalisé ou le stockage backend	Nom du pilote + "_" + dataLIF
`managementLIF`	Adresse IP d'une LIF de gestion de cluster ou de SVM. Un nom de domaine complet (FQDN) peut être spécifié. Il est possible de configurer l'utilisation d'adresses IPv6 si Trident a été installé avec l'option IPv6. Les adresses IPv6 doivent être définies entre crochets, comme `[28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]`. Pour une transition en douceur MetroCluster, consultez le Exemple MetroCluster. **REMARQUE** : Si vous utilisez les identifiants « vsadmin », `managementLIF` doit être celui de la SVM ; si vous utilisez les identifiants « admin », `managementLIF` doit être celui du cluster.	"10.0.0.1", "[2001:1234:abcd::fefe]"

Paramètre	Description	Défaut
`dataLIF`	Adresse IP de l'interface logique de protocole (LIF). Il est possible de configurer l'utilisation d'adresses IPv6 si Trident a été installé avec l'option IPv6. Les adresses IPv6 doivent être définies entre crochets, comme `[28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]`. **Ne pas spécifier pour iSCSI.** Trident utilise "Mappage LUN sélectif ONTAP" pour découvrir les LIF iSCSI nécessaires à l'établissement d'une session multi-chemin. Un avertissement est généré si `dataLIF` est explicitement défini. **Omettre pour MetroCluster.** Voir le Exemple MetroCluster.	Dérivé par le SVM
`svm`	Machine virtuelle de stockage à utiliser **Omettre pour MetroCluster.** Voir le Exemple MetroCluster.	Dérivé si un SVM `managementLIF` est spécifié
`useCHAP`	Utilisez CHAP pour authentifier iSCSI pour les pilotes SAN ONTAP [paramètre booléen]. Définissez sur `true` pour que Trident configure et utilise CHAP bidirectionnel comme authentification par défaut pour la SVM indiquée dans le backend. Consultez "Préparez-vous à configurer le backend 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. Obligatoire si `useCHAP=true`	""
`labels`	Ensemble d'étiquettes arbitraires au format JSON à appliquer aux volumes	""
`chapTargetInitiatorSecret`	Secret de l'initiateur de la cible CHAP. Obligatoire si `useCHAP=true`	""
`chapUsername`	Nom d'utilisateur entrant. Obligatoire si `useCHAP=true`	""
`chapTargetUsername`	Nom d'utilisateur cible. Obligatoire 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 d'autorité de certification de confiance. Facultatif. Utilisé pour l'authentification par certificat.	""
`username`	Nom d'utilisateur requis pour communiquer avec le cluster ONTAP. Utilisé pour l'authentification par identifiants. Pour l'authentification Active Directory, voir "Authentifier Trident auprès d'un SVM backend à l'aide des identifiants Active Directory".	""

Paramètre	Description	Défaut
`password`	Mot de passe requis pour communiquer avec le cluster ONTAP. Utilisé pour l'authentification par identifiants. Pour l'authentification Active Directory, voir "Authentifier Trident auprès d'un SVM backend à l'aide des identifiants Active Directory".	""
`svm`	Machine virtuelle de stockage à utiliser	Dérivé si un SVM `managementLIF` est spécifié
`storagePrefix`	Préfixe utilisé lors du provisionnement de nouveaux volumes dans la SVM. Ne peut pas être modifié ultérieurement. Pour mettre à jour ce paramètre, vous devrez créer un nouveau backend.	`trident`
`aggregate`	Agrégat pour le provisionnement (facultatif ; s'il est défini, il doit être attribué à la SVM). Pour le `ontap-nas-flexgroup` driver, cette option est ignorée. S'il n'est pas attribué, n'importe quel agrégat disponible peut être utilisé pour provisionner un FlexGroup volume. **REMARQUE** : Lorsqu'un agrégat est mis à jour dans SVM, il est automatiquement mis à jour dans Trident par interrogation de SVM, sans qu'il soit nécessaire de redémarrer le contrôleur Trident. Si vous avez configuré un agrégat spécifique dans Trident pour provisionner des volumes, si l'agrégat est renommé ou déplacé hors de la SVM, le backend passera en état d'échec dans Trident lors de l'interrogation de l'agrégat SVM. Vous devez soit changer l'agrégat pour un qui est présent sur la SVM, soit le supprimer complètement pour remettre le backend en ligne. **Ne pas spécifier pour les systèmes ASA r2**.	""
`limitAggregateUsage`	L'approvisionnement échoue si l'utilisation dépasse ce pourcentage. Si vous utilisez un Amazon FSx for NetApp ONTAP backend, ne spécifiez pas `limitAggregateUsage`. Les `fsxadmin` et `vsadmin` fournis ne contiennent pas les autorisations requises pour récupérer l'utilisation agrégée et la limiter à l'aide de Trident. **Ne pas spécifier pour les systèmes ASA r2**.	"" (non appliqué par défaut)

Paramètre	Description	Défaut
`limitVolumeSize`	L'approvisionnement échoue si la taille du volume demandée dépasse 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`	Options de débogage à utiliser lors du dépannage. Exemple, {"api":false, "method":true} n'utilisez-les que si vous effectuez un dépannage et avez besoin d'un journal détaillé.	`null`
`useREST`	Paramètre booléen à utiliser pour les API REST ONTAP. `` `useREST` Lorsqu'il est défini 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 ou une version ultérieure. De plus, le rôle de connexion ONTAP utilisé doit avoir accès à l'application `ontapi`. Cela est satisfait par les rôles prédéfinis `vsadmin` et `cluster-admin`. À partir de la version Trident 24.06 et ONTAP 9.15.1 ou version ultérieure, `useREST` est défini sur `true` par défaut ; modifiez `useREST` sur `false` pour utiliser les appels ONTAPI (ZAPI). `` `useREST` est entièrement compatible NVMe/TCP. **REMARQUE** : NVMe est pris en charge uniquement avec les API REST ONTAP et non avec ONTAPI (ZAPI). **Si spécifié, toujours définir sur `true` pour les systèmes ASA r2**.	`true` pour ONTAP 9.15.1 ou version ultérieure, sinon `false`.

Paramètre	Description	Défaut
`sanType`	Utilisez pour sélectionner `iscsi` pour iSCSI, `nvme` pour NVMe/TCP ou `fcp` pour SCSI sur Fibre Channel (FC).	`iscsi` si vide
`formatOptions`	Utilisez `formatOptions` pour spécifier des arguments de ligne de commandes pour la commande `mkfs`, qui seront appliqués chaque fois qu'un volume est formaté. Cela vous permet de formater le volume selon vos préférences. Assurez-vous de spécifier les formatOptions similaires à celles des options de la commande mkfs, à l'exclusion du chemin du périphérique. Exemple : « -E nodiscard » **Pris en charge pour `ontap-san` et `ontap-san-economy` pilotes avec le protocole iSCSI. En outre, pris en charge pour les systèmes ASA r2 lors de l'utilisation des protocoles iSCSI et NVMe/TCP.**	
`limitVolumePoolSize`	Taille maximale requise de FlexVol lors de l'utilisation de LUN dans le backend ontap-san-economy.	"" (non appliqué par défaut)
`denyNewVolumePools`	Limite `ontap-san-economy` les backends à la création de nouveaux volumes FlexVol® pour contenir leurs LUN. Seuls les FlexVol® préexistants sont utilisés pour le provisionnement de nouveaux PV.	

Recommandations d'utilisation de formatOptions

Trident recommande les options suivantes pour accélérer le processus de mise en forme :

- **-E nodiscard (ext3, ext4) :** Ne pas tenter de supprimer des blocs lors de la création du système de fichiers (la suppression initiale de blocs est utile sur les disques SSD et les systèmes de stockage à provisionnement fin). Cette option remplace l'option obsolète « -K » et s'applique aux systèmes de fichiers ext3 et ext4.
- **-K (xfs) :** Ne pas tenter de supprimer des blocs lors de la création du système de fichiers (mkfs). Cette option s'applique au système de fichiers xfs.

Authentifier Trident auprès d'un SVM backend à l'aide des identifiants Active Directory

Vous pouvez configurer Trident pour qu'il s'authentifie auprès d'une SVM backend à l'aide d'identifiants Active Directory (AD). Avant qu'un compte AD puisse accéder à la SVM, vous devez configurer l'accès des contrôleurs de domaine AD au cluster ou à la SVM. Pour l'administration du cluster avec un compte AD, vous devez créer un tunnel de domaine. Consultez "Configurer l'accès au contrôleur de domaine Active Directory dans ONTAP" pour plus de détails.

étapes

1. Configurer les paramètres du système de noms de domaine (DNS) pour une 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 la 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 afin de gérer le cluster ou la 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 backend de Trident, définissez les paramètres `username` et `password` sur le nom d'utilisateur ou de groupe AD et le mot de passe, respectivement.

Options de configuration backend pour le provisionnement des volumes

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

Paramètre	Description	Défaut
`spaceAllocation`	Allocation d'espace pour les LUNs	"true" **Si spécifié, définissez sur `true` pour les systèmes ASA r2**.
`spaceReserve`	Mode de réservation d'espace : « aucun » (fin) ou « volume » (épais). **À définir sur `none` pour les systèmes ASA r2**.	"none"
`snapshotPolicy`	Stratégie d'instantané à utiliser. **À définir sur `none` pour les systèmes ASA r2**.	"none"
`qosPolicy`	Groupe de règles QoS à attribuer aux volumes créés. Choisissez l'un de qosPolicy ou adaptiveQosPolicy par pool de stockage/backend. L'utilisation des groupes de règles QoS avec Trident requiert 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 individuellement à chaque composant. Un groupe de règles QoS partagé impose une limite au débit total de toutes les charges de travail.	""
`adaptiveQosPolicy`	Groupe de règles QoS adaptatives à attribuer aux volumes créés. Choisissez l'un des deux, qosPolicy ou adaptiveQosPolicy, par pool de stockage/backend	""
`snapshotReserve`	Pourcentage du volume réservé aux 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	"false"

Paramètre	Description	Défaut
`encryption`	Activez le chiffrement de volume NetApp (NVE) sur le nouveau volume ; la valeur par défaut est `false`. NVE doit être sous licence et activé sur le cluster pour utiliser cette option. Si NAE est activé sur le backend, tout volume provisionné dans Trident sera activé pour NAE. Pour plus d'informations, consultez : "Comment Trident fonctionne avec NVE et NAE".	"false" **Si spécifié, définissez sur `true` pour les systèmes ASA r2**.
`luksEncryption`	Activez le chiffrement LUKS. Consultez "Utilisez Linux Unified Key Setup (LUKS)".	"" **Définir sur `false` pour les systèmes ASA r2**.
`tieringPolicy`	Politique de hiérarchisation à utiliser « none » **Ne pas spécifier pour les systèmes ASA r2**.	
`nameTemplate`	Modèle pour créer des noms de volumes 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` driver, Trident ajoute 10 % de capacité supplémentaire à la FlexVol pour intégrer les métadonnées LUN. La LUN sera provisionnée avec la taille exacte demandée par l'utilisateur dans le PVC. Trident ajoute 10 % à la FlexVol (affiché comme taille disponible dans ONTAP). Les utilisateurs obtiendront désormais la capacité utilisable qu'ils ont demandée. Cette modification empêche également les LUN de devenir en lecture seule, sauf si l'espace disponible est entièrement utilisé. Ceci ne s'applique pas à ontap-san-economy.

Pour les backends qui définissent `snapshotReserve`, Trident calcule la taille des volumes comme suit :

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

Le chiffre 1,1 correspond aux 10 % supplémentaires que Trident ajoute au FlexVol pour prendre en compte les métadonnées LUN. Pour `snapshotReserve` = 5 %, et une demande PVC = 5 Gio, la taille totale du volume est de 5,79 Gio et la taille disponible est de 5,5 Gio. La commande `volume show` 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 minimale

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.

REMARQUE	Si vous utilisez Amazon FSx sur NetApp ONTAP avec Trident, NetApp recommande de spécifier les noms DNS des LIF au lieu des adresses IP.

Exemple SAN ONTAP

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

```
---
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 backend pour éviter d'avoir à mettre à jour manuellement la définition du backend après le basculement et le retour en arrière pendant "Réplication et récupération de SVM".

Pour une transition et un retour en arrière sans heurt, spécifiez le SVM en utilisant `managementLIF` et omettez 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 par certificat

Dans cet exemple de configuration de base `clientCertificate`, `clientPrivateKey`, et `trustedCACertificate` (facultatif, si vous utilisez une autorité de certification de confiance) sont renseignés dans `backend.json` et prennent respectivement les valeurs encodées en base64 du certificat client, de la clé privée et du certificat de l'autorité de certification de confiance.

```
---
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` défini sur `true`.

Exemple ONTAP SAN CHAP

```
---
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 d'économie ONTAP SAN CHAP

```
---
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

Vous devez disposer d'une SVM configurée avec NVMe sur votre backend ONTAP. Il s'agit d'une configuration 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 disposer d'une SVM configurée avec FC sur votre backend ONTAP. Il s'agit d'une configuration de base du backend 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 du backend avec nameTemplate

```
---
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}}"
```

formatOptions exemple 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 backends avec pools virtuels

Dans ces exemples de fichiers de définition de backend, des valeurs par défaut spécifiques sont définies pour tous les pools de stockage, comme `spaceReserve` à aucun, `spaceAllocation` à faux, et `encryption` à faux. Les pools virtuels sont définis dans la section storage.

Trident définit les étiquettes de provisionnement dans le champ « Commentaires ». Les commentaires sont définis sur le volume FlexVol. Trident copie toutes les étiquettes présentes sur un pool virtuel vers le volume de stockage lors du provisionnement. Pour plus de simplicité, les administrateurs de stockage peuvent définir des étiquettes par pool virtuel et regrouper les volumes par étiquette.

Dans ces exemples, certains pools de stockage définissent leurs propres `spaceReserve`, `spaceAllocation`, et `encryption` valeurs, et certains pools remplacent les valeurs par défaut.

Exemple 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"
```

Map backends vers StorageClasses

Les définitions suivantes de StorageClass se réfèrent à Exemples de backends avec pools virtuels. En utilisant le champ `parameters.selector`, chaque StorageClass indique quels pools virtuels peuvent être utilisés pour héberger un volume. Le volume possédera les aspects définis dans le pool virtuel choisi.

- Le `protection-gold` StorageClass sera associé au premier pool virtuel dans le `ontap-san` backend. Il s'agit du seul pool offrant une protection de niveau or.

```
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 associé aux deuxième et troisième pools virtuels dans `ontap-san` backend. 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 correspondra au troisième pool virtuel dans le `ontap-san-economy` backend. Il s'agit du seul pool offrant une configuration de pool de stockage pour l'app 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 deuxième pool virtuel dans `ontap-san` 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 associé au troisième pool virtuel dans le `ontap-san` backend et au quatrième pool virtuel dans le `ontap-san-economy` backend. Ce sont les seuls pools proposant 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` StorageClass sera associé au `testAPP` pool virtuel dans le `ontap-san` pilote avec `sanType: nvme`. C'est le seul pool offrant `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 déterminera quel pool virtuel est sélectionné et s'assurera que l'exigence de stockage est respectée.

Pilotes ONTAP NAS

Présentation du pilote ONTAP NAS

Découvrez comment configurer un backend ONTAP avec les pilotes NAS ONTAP et Cloud Volumes ONTAP.

Détails du pilote ONTAP NAS

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

Pilote	Protocole	volumeMode	Modes d'accès pris en charge	Systèmes de fichiers pris en charge
`ontap-nas`	NFS SMB	Système de fichiers	RWO, ROX, RWX, RWOP	"", `nfs`, `smb`
`ontap-nas-economy`	NFS SMB	Système de fichiers	RWO, ROX, RWX, RWOP	"", `nfs`, `smb`

Pilote	Protocole	volumeMode	Modes d'accès pris en charge	Systèmes de fichiers pris en charge
`ontap-nas-flexgroup`	NFS SMB	Système de fichiers	RWO, ROX, RWX, RWOP	"", `nfs`, `smb`

AVERTISSEMENT

- Utilisez `ontap-san-economy` uniquement si le nombre d'utilisations du volume persistant doit être supérieur à "limites de volume ONTAP prises en charge".
- Utilisez `ontap-nas-economy` uniquement si le nombre d'utilisations de volumes persistants est censé être supérieur à "limites de volume ONTAP prises en charge" et que le pilote `ontap-san-economy` ne peut pas être utilisé.
- N'utilisez pas `ontap-nas-economy` si vous prévoyez un besoin de protection des données, de reprise après sinistre ou de mobilité.
- NetApp ne recommande pas d'utiliser la croissance automatique FlexVol® dans tous les pilotes ONTAP, sauf ontap-san. En guise de solution de contournement, Trident prend en charge l'utilisation de la réserve de snapshots et adapte les volumes FlexVol® en conséquence.

Autorisations de l'utilisateur

Trident doit être exécuté en tant qu'administrateur ONTAP ou SVM, généralement en utilisant l' `admin`utilisateur cluster ou un `vsadmin`utilisateur SVM, ou un utilisateur avec un nom différent ayant le même rôle.

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

REMARQUE

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

Bien qu'il soit possible de créer un rôle plus restrictif au sein d'ONTAP qu'un pilote Trident peut utiliser, nous ne le recommandons pas. La plupart des nouvelles versions de Trident feront appel à des API supplémentaires dont il faudrait tenir compte, ce qui rend les mises à niveau difficiles et sujettes aux erreurs.

Préparez-vous à configurer un backend 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 "NetApp système de stockage 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	Seul le `ontap-nas` driver (avec le protocole NFS) est pris en charge pour les systèmes AFX; le protocole SMB n'est pas pris en charge.

Exigences

- Pour tous les backends ONTAP, Trident exige qu'au moins un agrégat soit affecté 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` driver et une classe Bronze qui utilise le `ontap-nas-economy` driver.
- Tous vos nœuds de travail Kubernetes doivent disposer des outils NFS appropriés. Consultez "ici" pour plus de détails.
- Trident prend uniquement en charge les volumes SMB montés sur des pods exécutés sur des nœuds Windows. Consultez Préparez-vous à provisionner des volumes SMB pour plus de détails.

Authentifier le backend ONTAP

Trident propose deux modes d'authentification pour un backend ONTAP.

- Basé sur les identifiants : ce mode requiert des autorisations suffisantes sur le backend ONTAP. Il est recommandé d'utiliser un compte associé à un rôle de connexion de sécurité prédéfini, tel que `admin` ou `vsadmin` afin de garantir une compatibilité maximale avec les versions ONTAP.
- Authentification par certificat : ce mode requiert l'installation d'un certificat sur le backend pour que Trident puisse communiquer avec un ONTAP cluster. Dans ce cas, la définition du backend doit contenir les valeurs encodées en Base64 du certificat client, de la clé et, si utilisé (recommandé), du certificat de l'autorité de certification de confiance.

Vous pouvez mettre à jour les backends existants pour basculer entre les méthodes basées sur les identifiants et celles basées sur les certificats. Cependant, une seule méthode d'authentification est prise en charge à la fois. Pour passer à une autre méthode d'authentification, vous devez supprimer la méthode existante de la configuration du backend.

AVERTISSEMENT	Si vous tentez de fournir **à la fois des informations d'identification et des certificats**, la création du backend échouera 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 identifiants

Trident requiert les identifiants d'un administrateur au niveau SVM ou au niveau cluster pour communiquer avec le backend ONTAP. Il est recommandé d'utiliser des rôles standard prédéfinis tels que `admin` ou `vsadmin`. Cela garantit la compatibilité avec les futures versions d'ONTAP qui pourraient exposer des API de fonctionnalités à utiliser par les futures versions de Trident. Un rôle de connexion de sécurité personnalisé peut être créé et utilisé avec Trident, mais cela n'est pas recommandé.

Voici un exemple de définition de backend :

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"
    }
}
```

Il est important de noter que la définition du backend est le seul endroit où les identifiants sont stockés en clair. Après la création du backend, les noms d'utilisateur et mots de passe sont encodés avec Base64 et stockés comme secrets Kubernetes. La création/la mise à jour d'un backend est la seule étape nécessitant la connaissance des identifiants. En tant que telle, il s'agit d'une opération réservée à l'administrateur Kubernetes/stockage.

Activer l'authentification par certificat

Les nouveaux et les anciens backends peuvent utiliser un certificat et communiquer avec le backend ONTAP. Trois paramètres sont requis dans la définition du backend.

- clientCertificate: Valeur codée en Base64 du certificat client.
- clientPrivateKey : valeur encodée en Base64 de la clé privée associée.
- trustedCACertificate : valeur encodée en Base64 du certificat CA de confiance. Si vous utilisez un CA de confiance, ce paramètre doit être fourni. Cela peut être ignoré si aucun CA de confiance n'est utilisé.

Un workflow typique implique 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 à authentifier.

```
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. Cette opération peut déjà être effectuée par l'administrateur de stockage. Ignorez si aucune autorité de certification de confiance 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 (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. Confirmez que le rôle de connexion de sécurité ONTAP prend en charge `cert` la méthode 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 du certificat généré. Remplacez <ONTAP Management LIF> et <vserver name> par l'adresse IP de la LIF de gestion et le nom SVM. Vous devez vous assurer que le LIF a sa stratégie 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 d'autorité de certification 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 backend en utilisant les valeurs obtenues à 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 |
+------------+----------------+--------------------------------------
+--------+---------+
```

Mettez à jour les méthodes d'authentification ou faites pivoter les identifiants

Vous pouvez mettre à jour un backend existant pour utiliser une méthode d'authentification différente ou pour faire pivoter ses identifiants. Cela fonctionne dans les deux sens : les backends qui utilisent un nom d'utilisateur/mot de passe peuvent être mis à jour pour utiliser des certificats ; les backends qui utilisent des certificats peuvent être mis à jour pour utiliser un nom d'utilisateur/mot de passe. Pour ce faire, vous devez supprimer la méthode d'authentification existante et ajouter la nouvelle méthode d'authentification. Ensuite, utilisez le fichier backend.json mis à jour contenant les paramètres requis pour 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 du renouvellement 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 backend. Lors du renouvellement des certificats, plusieurs certificats peuvent être associés à l'utilisateur. Le backend est ensuite mis à jour pour utiliser le nouveau certificat, après quoi l'ancien certificat peut être supprimé du cluster ONTAP.

La mise à jour d'un backend n'interrompt pas l'accès aux volumes déjà créés et n'affecte pas les connexions de volumes établies ultérieurement. Une mise à jour réussie du backend indique que Trident peut communiquer avec le backend ONTAP et gérer les futures opérations sur les volumes.

Créer un rôle ONTAP personnalisé pour Trident

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

créé pour effectuer les opérations.

Reportez-vous à "Générateur de rôles personnalisés Trident" pour plus d'informations sur la création de rôles personnalisés Trident.

Utilisation de l'interface de ligne de commande ONTAP

1. Créez un nouveau 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. Associez le rôle à l'utilisateur :

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

Utilisation de System Manager

Effectuez les étapes suivantes dans ONTAP System Manager :

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

      (Ou) Pour créer un rôle personnalisé au niveau de la SVM, sélectionnez **Storage > Storage VMs > `required SVM`> Paramètres > Utilisateurs et rôles**.

   b. Sélectionnez l'icône flèche (→) à côté de **Users and Roles**.
   c. Sélectionnez **+Add** sous **Roles**.
   d. Définissez les règles du rôle et cliquez sur **Save**.
2. **Associez le rôle à l'utilisateur Trident** : + Effectuez les étapes suivantes sur la page **Utilisateurs et rôles** :
   a. Sélectionnez l'icône Ajouter **+** sous **Users**.
   b. Sélectionnez le nom d'utilisateur requis, puis sélectionnez un rôle dans le menu déroulant pour **Rôle**.
   c. Cliquez sur **Enregistrer**.

Reportez-vous aux pages suivantes pour plus d'informations :

- "Rôles personnalisés pour l'administration d'ONTAP" ou "Définir des rôles personnalisés"
- "Travailler avec les rôles et les utilisateurs"

Gérer les règles d'export NFS

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

Trident propose deux options lors de l'utilisation des règles d'export :

- Trident peut gérer dynamiquement la règle d'export ; dans ce mode de fonctionnement, l'administrateur de stockage spécifie une liste de blocs CIDR représentant les adresses IP admissibles. Trident ajoute automatiquement à la règle d'export les adresses IP des nœuds concernés qui appartiennent à ces plages lors de la publication. Sinon, si aucun CIDR n'est spécifié, toutes les adresses IP unicast à portée globale trouvées sur le nœud auquel le volume est publié seront ajoutées à la règle d'export.
- Les administrateurs de stockage peuvent créer une règle d'export et y ajouter des règles manuellement. Trident utilise la règle d'export par défaut, sauf si un autre nom de règle d'export est spécifié dans la configuration.

Gérer dynamiquement les règles d'export

Trident offre la possibilité de gérer dynamiquement les règles d'export pour les backends ONTAP. Cela donne à l'administrateur de stockage la possibilité de spécifier un espace d'adresses autorisé pour les adresses IP des nœuds de travail, plutôt que de définir manuellement des règles explicites. Cela simplifie grandement la gestion des règles d'export ; les modifications apportées à la règle d'export 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 de travail qui montent des volumes et dont les adresses IP se trouvent dans la plage spécifiée, ce qui permet une gestion fine et automatisée.

REMARQUE	N'utilisez pas la traduction d'adresses réseau (NAT) lorsque vous utilisez des règles d'export dynamiques. Avec la NAT, le contrôleur de stockage voit l'adresse NAT du frontal et non l'adresse IP réelle de l'hôte, donc l'accès sera refusé si aucune correspondance n'est trouvée dans les règles d'export.

Exemple

Deux options de configuration doivent être utilisées. Voici un exemple de définition de backend :

```
---
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 racine de votre SVM possède une règle d'export précédemment créée avec une règle d'export autorisant le bloc CIDR du nœud (par exemple, la règle d'export par défaut). Suivez toujours la bonne pratique recommandée par NetApp pour dédier une SVM à Trident.

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

- `autoExportPolicy` est défini sur `true`. Cela indique que Trident crée une règle d'export pour chaque volume provisionné avec ce backend pour la `svm1` SVM et gère l'ajout et la suppression de règles à l'aide de blocs d'adresses `autoexportCIDRs`. Jusqu'à ce qu'un volume soit attaché à un nœud, le volume utilise une règle d'export vide sans aucune règle afin d'empêcher tout accès non désiré à ce volume. Lorsqu'un volume est publié sur un nœud, Trident crée une règle d'export portant le même nom que le qtree sous-jacent contenant l'adresse IP du nœud dans le bloc CIDR spécifié. Ces adresses IP seront également ajoutées à la règle d'export utilisée par le volume FlexVol parent.
  - Par exemple :
    - UUID de backend 403b5326-8482-40db-96d0-d83fb3f4daec
    - `autoExportPolicy` défini sur `true`
    - préfixe de stockage `trident`
    - UUID PVC a79bcf5f-7b6d-4a40-9876-e2551f159c1c
    - Le qtree nommé trident_pvc_a79bcf5f_7b6d_4a40_9876_e2551f159c1c crée une règle d'export pour le FlexVol nommé `trident-403b5326-8482-40db96d0-d83fb3f4daec`, une règle d'export pour le qtree nommé `trident_pvc_a79bcf5f_7b6d_4a40_9876_e2551f159c1c`, et une règle d'export vide nommée `trident_empty` sur la SVM. Les règles de la règle d'export du FlexVol seront un sur-ensemble de toutes les règles contenues dans les règles d'export du qtree. La règle d'export vide sera réutilisée par tous les volumes qui ne sont pas attachés.
- `autoExportCIDRs` contient une liste de blocs d'adresses. Ce champ est facultatif et il est défini par défaut sur ["0.0.0.0/0", "::/0"]. S'il n'est pas défini, Trident ajoute toutes les adresses unicast à portée globale trouvées sur les nœuds de travail avec des publications.

Dans cet exemple, l' `192.168.0.0/24`espace d'adresses est fourni. Cela indique que les adresses IP des nœuds Kubernetes qui se trouvent dans cette plage d'adresses avec des publications seront ajoutées à la règles d'export que Trident crée. Lorsque Trident enregistre un nœud sur lequel il s'exécute, il récupère les adresses IP du nœud et les vérifie par rapport aux blocs d'adresses fournis dans `autoExportCIDRs. Au moment de la publication, après avoir filtré les adresses IP, Trident crée les règles d'export pour les adresses IP clientes du nœud auquel il publie.

Vous pouvez mettre à jour `autoExportPolicy` et `autoExportCIDRs` pour les backends après leur création. Vous pouvez ajouter de nouveaux CIDR pour un backend géré automatiquement ou supprimer les CIDR existants. Faites attention lors de la suppression de CIDR afin de garantir que les connexions existantes ne soient pas interrompues. Vous pouvez également choisir de désactiver `autoExportPolicy` pour un backend et revenir à une règle d'export créée manuellement. Cela nécessitera de définir le paramètre `exportPolicy` dans la configuration de votre backend.

Après que Trident a créé ou mis à jour un backend, vous pouvez vérifier le backend en utilisant `tridentctl` ou le 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 règles d'export afin de supprimer les règles d'accès correspondantes au nœud. En supprimant cette adresse IP de nœud des règles d'export des backends gérés, Trident empêche les montages non autorisés, sauf si cette adresse IP est réutilisée par un nouveau nœud dans le cluster.

Pour les backends existants, la mise à jour du backend avec `tridentctl update backend` garantit que Trident gère automatiquement les règles d'export. Cela crée deux nouvelles règles d'export nommées d'après l'UUID du backend et le nom du qtree lorsqu'elles sont nécessaires. Les volumes présents sur le backend utiliseront les nouvelles règles d'export après avoir été démontés puis remontés.

REMARQUE	La suppression d'un backend avec des règles d'export gérées automatiquement supprimera la règle d'export créée dynamiquement. Si le backend est recréé, il est traité comme un nouveau backend et cela entraînera la création d'une nouvelle règle d'export.

Si l'adresse IP d'un nœud en production est modifiée, vous devez redémarrer le pod Trident sur ce nœud. Trident mettra alors à jour la règle d'export pour les backends qu'il gère afin de refléter ce changement d'adresse IP.

Préparez-vous à provisionner des volumes SMB

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

AVERTISSEMENT	Vous devez configurer les protocoles NFS et SMB/CIFS sur la SVM pour créer un `ontap-nas-economy` volume SMB pour les clusters ONTAP sur site. L'absence de configuration de l'un ou l'autre de ces protocoles entraînera 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.

- Un cluster Kubernetes avec un nœud contrôleur Linux et au moins un nœud de travail Windows exécutant Windows Server 2022. Trident prend uniquement en charge les volumes SMB montés sur des pods exécutés sur des nœuds Windows.
- Au moins un secret Trident contenant vos identifiants Active Directory. Pour générer le 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`, reportez-vous à "GitHub : CSI Proxy" ou "GitHub: CSI Proxy pour Windows" pour les nœuds Kubernetes exécutés sous Windows.

Étapes

1. Pour ONTAP sur site, vous pouvez éventuellement créer un partage SMB ou Trident peut 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 : soit à l'aide du "Microsoft Management Console" Shared Folders snap-in, soit 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 commande ONTAP :

   a. Si nécessaire, créez la structure du chemin d'accès du répertoire pour le partage.

      La `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 à la SVM spécifiée :

      ```
      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 "Créer un partage SMB" pour plus de détails.

2. Lors de la création du backend, vous devez configurer les éléments suivants pour spécifier les volumes SMB. Pour toutes les options de configuration du backend FSx pour ONTAP, reportez-vous à "Options de configuration et exemples pour FSx for 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 pour permettre à Trident de créer le partage SMB ; ou vous pouvez laisser le paramètre vide pour empêcher l'accès partagé aux volumes. Ce paramètre est facultatif pour ONTAP sur site. Ce paramètre est obligatoire pour Amazon FSx for ONTAP backends et ne peut pas être vide.	`smb-share`
`nasType`	**Doit être défini sur `smb`.** Si nul, 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` or `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 de `ontap-nas` et `ontap-nas-economy` serveurs backend. 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 des listes de contrôle d'accès (ACL).

Points à retenir

- L'importation `ontap-nas-economy` de volumes n'est pas prise en charge.
- Seuls les clones en lecture seule sont pris en charge pour `ontap-nas-economy` volumes.
- Si le protocole SMB sécurisé 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.
- La liste de contrôle d'accès (ACL) de partage SMB spécifiée dans l'annotation du PVC cloné aura priorité sur celles du PVC source.
- Veillez à fournir des utilisateurs AD valides lors de l'activation du protocole SMB sécurisé. Les utilisateurs non valides ne seront pas ajoutés à l'ACL.
- Si vous fournissez le même utilisateur AD dans le backend, la storage class et le PVC avec des autorisations différentes, la priorité des autorisations sera : PVC, storage class, puis backend.
- Le protocole SMB sécurisé est pris en charge pour `ontap-nas` les importations de volumes gérés et n'est pas applicable aux importations de volumes non gérés.

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 l' `trident.netapp.io/smbShareAdUser`annotation à la classe de stockage pour activer SMB sécurisé sans faute. La valeur utilisateur spécifiée pour l'annotation `trident.netapp.io/smbShareAdUser` doit être identique au nom d'utilisateur indiqué 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 un 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 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 "NetApp AFX systèmes de stockage". 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	Seul le `ontap-nas` driver (avec le protocole NFS) est pris en charge pour les systèmes NetApp AFX ; le protocole SMB n'est pas pris en charge.

Options de configuration du backend

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	Défaut
`version`		Toujours 1

Paramètre	Description	Défaut
`storageDriverName`	Nom du pilote de stockage **REMARQUE** Pour les systèmes NetApp AFX, seul `ontap-nas` est pris en charge.	`ontap-nas`, `ontap-nas-economy`, ou `ontap-nas-flexgroup`
`backendName`	Nom personnalisé ou le stockage backend	Nom du pilote + "_" + dataLIF
`managementLIF`	Adresse IP d'un cluster ou d'une LIF de gestion SVM. Un nom de domaine complet (FQDN) peut être spécifié. Il est possible de configurer l'utilisation d'adresses IPv6 si Trident a été installé avec l'option IPv6. Les adresses IPv6 doivent être définies entre crochets, comme `[28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]`. Pour une transition en douceur MetroCluster, consultez le Exemple MetroCluster.	"10.0.0.1", "[2001:1234:abcd::fefe]"
`dataLIF`	Adresse IP de l'interface logique de protocole (LIF). 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 complet (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 la configuration initiale. Consultez . Il est possible de configurer l'utilisation d'adresses IPv6 si Trident a été installé avec l'option IPv6. Les adresses IPv6 doivent être définies entre crochets, comme `[28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]`. **Omettre pour MetroCluster.** Voir le Exemple MetroCluster.	Adresse spécifiée ou dérivée de la SVM, si non spécifiée (non recommandé)
`svm`	Machine virtuelle de stockage à utiliser **Omettre pour MetroCluster.** Voir le Exemple MetroCluster.	Dérivé si un SVM `managementLIF` est spécifié
`autoExportPolicy`	Activer la création et la mise à jour automatiques des règles d'export [Booléen]. En utilisant les options `autoExportPolicy` et `autoExportCIDRs`, Trident peut gérer les règles d'export automatiquement.	false
`autoExportCIDRs`	Liste des CIDR pour filtrer les adresses IP des nœuds Kubernetes lorsque `autoExportPolicy` est activé. En utilisant les options `autoExportPolicy` et `autoExportCIDRs`, Trident peut gérer les règles d'export automatiquement.	["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	Défaut
`clientPrivateKey`	Valeur encodée en Base64 de la clé privée du client. Utilisé pour l'authentification par certificat	""
`trustedCACertificate`	Valeur encodée en Base64 du certificat d'autorité de certification de confiance. Facultatif. Utilisé pour l'authentification par certificat	""
`username`	Nom d'utilisateur pour se connecter au cluster/SVM. Utilisé pour l'authentification par identifiants. Pour l'authentification Active Directory, voir "Authentifier Trident auprès d'un SVM backend à l'aide des identifiants Active Directory".	
`password`	Mot de passe pour se connecter au cluster/SVM. Utilisé pour l'authentification par identifiants. Pour l'authentification Active Directory, voir "Authentifier Trident auprès d'un SVM backend à l'aide des identifiants Active Directory".	
`storagePrefix`	Préfixe utilisé lors du provisionnement de nouveaux volumes dans la SVM. Ne peut pas être modifié après sa définition **REMARQUE** Lors de l'utilisation d'ontap-nas-economy et d'un storagePrefix de 24 caractères ou plus, les qtrees n'auront pas le préfixe de stockage intégré, bien qu'il soit présent dans le nom du volume.	"Trident"

Paramètre	Description	Défaut
`aggregate`	Agrégat pour le provisionnement (facultatif ; s'il est défini, il doit être attribué à la SVM). Pour le `ontap-nas-flexgroup` driver, cette option est ignorée. S'il n'est pas attribué, n'importe quel agrégat disponible peut être utilisé pour provisionner un FlexGroup volume. **REMARQUE** : Lorsqu'un agrégat est mis à jour dans SVM, il est automatiquement mis à jour dans Trident par interrogation de SVM, sans qu'il soit nécessaire de redémarrer le contrôleur Trident. Si vous avez configuré un agrégat spécifique dans Trident pour provisionner des volumes, si l'agrégat est renommé ou déplacé hors de la SVM, le backend passera en état d'échec dans Trident lors de l'interrogation de l'agrégat SVM. Vous devez soit changer l'agrégat pour un qui est présent sur la SVM, soit le supprimer complètement pour remettre le backend en ligne. **Ne pas spécifier pour les systèmes de stockage AFX.**	""
`limitAggregateUsage`	L'approvisionnement échoue 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	Défaut
flexgroupAggregateList	Liste des agrégats à provisionner (facultatif ; si défini, doit être affecté au SVM). Tous les agrégats affectés au SVM sont utilisés pour provisionner un FlexGroup volume. Pris en charge par le pilote de stockage **ontap-nas-flexgroup**. **REMARQUE** : Lorsque la liste d'agrégats est mise à jour dans SVM, la liste est mise à jour automatiquement dans Trident par interrogation de SVM, sans qu'il soit nécessaire de redémarrer le contrôleur Trident. Lorsque vous avez configuré une liste d'agrégats spécifique dans Trident pour le provisionnement de volumes, si la liste d'agrégats est renommée ou déplacée hors de SVM, le backend passera à l'état d'échec dans Trident lors de l'interrogation de l'agrégat SVM. Vous devez soit modifier la liste d'agrégats pour en choisir une présente sur le SVM, soit la supprimer complètement afin de remettre le backend en ligne.	""
`limitVolumeSize`	L'approvisionnement échoue si la taille du volume demandée dépasse cette valeur.	"" (non appliqué par défaut)
`debugTraceFlags`	Options de débogage à utiliser lors du dépannage. Exemple, {"api":false, "method":true} Ne pas utiliser `debugTraceFlags` sauf si vous effectuez un dépannage et avez besoin d'un journal détaillé.	null
`nasType`	Configurez la création de volumes NFS ou SMB. Les options sont `nfs`, `smb` ou null. La valeur null correspond par défaut à des volumes NFS. **Si spécifié, définissez toujours sur `nfs` pour les systèmes de stockage AFX**.	`nfs`

Paramètre	Description	Défaut
`nfsMountOptions`	Liste d'options de montage NFS séparées par des virgules. Les options de montage pour les volumes persistants Kubernetes sont généralement spécifiées dans les classes de stockage, mais si aucune option de montage n'est spécifiée dans une classe de stockage, Trident utilisera les options de montage spécifiées dans le fichier de configuration du backend de stockage. Si aucune option de montage n'est spécifiée ni dans la classe de stockage ni dans le fichier de configuration, Trident n'appliquera aucune option de montage au volume persistant associé.	""
`qtreesPerFlexvol`	Nombre maximal de Qtrees par FlexVol, 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 pour permettre à Trident de créer le partage SMB ; ou vous pouvez laisser le paramètre vide pour empêcher l'accès partagé aux volumes. Ce paramètre est facultatif pour ONTAP sur site. Ce paramètre est obligatoire pour Amazon FSx for ONTAP backends et ne peut pas être vide.	`smb-share`
`useREST`	Paramètre booléen à utiliser pour les API REST ONTAP. `useREST` Lorsqu'il est défini 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 ou une version ultérieure. De plus, le rôle de connexion ONTAP utilisé doit avoir accès à l'application `ontapi`. Cela est satisfait par les rôles prédéfinis `vsadmin` et `cluster-admin`. À partir de la version Trident 24.06 et ONTAP 9.15.1 ou version ultérieure, `useREST` est défini sur `true` par défaut ; modifiez `useREST` sur `false` pour utiliser les appels ONTAPI (ZAPI). **Si spécifié, définissez toujours sur `true` pour les systèmes de stockage AFX**.	`true` pour ONTAP 9.15.1 ou version ultérieure, sinon `false`.
`limitVolumePoolSize`	Taille maximale requise de FlexVol lors de l'utilisation de Qtrees dans le backend ontap-nas-economy.	"" (non appliqué par défaut)
`denyNewVolumePools`	Empêche `ontap-nas-economy` les serveurs backend de créer de nouveaux volumes FlexVol pour y stocker leurs Qtrees. Seuls les FlexVol® préexistants sont utilisés pour le provisionnement de nouveaux PV.	

Paramètre	Description	Défaut
adAdminUser	Utilisateur ou groupe d'utilisateurs administrateur Active Directory disposant d'un 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 backend pour le provisionnement des volumes

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

Paramètre	Description	Défaut
spaceAllocation	Allocation d'espace pour les Qtrees	"true"
spaceReserve	Mode de réservation d'espace ; "none" (fin) ou "volume" (épais)	"none"
snapshotPolicy	Stratégie de snapshot à utiliser	"none"
qosPolicy	Groupe de règles QoS à attribuer aux volumes créés. Choisissez l'un des deux, qosPolicy ou adaptiveQosPolicy, par pool de stockage/backend	""
adaptiveQosPolicy	Groupe de règles QoS adaptatives à attribuer aux volumes créés. Choisissez l'un de qosPolicy ou adaptiveQosPolicy par pool de stockage/backend. Non pris en charge par ontap-nas-economy.	""
snapshotReserve	Pourcentage du volume réservé aux instantanés	"0" si `snapshotPolicy` est "aucun", sinon ""
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`. NVE doit être sous licence et activé sur le cluster pour utiliser cette option. Si NAE est activé sur le backend, tout volume provisionné dans Trident sera activé pour NAE. Pour plus d'informations, consultez : "Comment Trident fonctionne avec NVE et NAE".	"false"
tieringPolicy	Politique de hiérarchisation à utiliser « none »	
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	Règles d'export à utiliser	"default"

Paramètre	Description	Défaut
`securityStyle`	Style de sécurité pour les nouveaux volumes. NFS prend en charge `mixed` et `unix` styles de sécurité. SMB prend en charge `mixed` et `ntfs` styles de sécurité.	La valeur par défaut de NFS est `unix`. La valeur par défaut de SMB est `ntfs`.
`nameTemplate`	Modèle pour créer des noms de volumes personnalisés.	""

REMARQUE	L'utilisation des groupes de règles QoS avec Trident requiert 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 individuellement à chaque composant. Un groupe de règles QoS partagé impose une limite au 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 afin de garantir que le FlexVol soit correctement dimensionné avec le pourcentage de snapshotReserve et le PVC. Lorsque l'utilisateur demande un PVC, Trident crée le FlexVol initial avec plus d'espace en utilisant le nouveau calcul.

Ce calcul garantit que l'utilisateur reçoit l'espace inscriptible qu'il a demandé dans le PVC, et non moins que ce qu'il a demandé. Avant la version v21.07, lorsque l'utilisateur demandait un PVC (par exemple, 5 Gio), avec le snapshotReserve à 50 pour cent, il n'obtenait que 2,5 Gio d'espace inscriptible. Cela s'explique par le fait que ce que l'utilisateur demandait était le volume entier et `snapshotReserve` est un pourcentage de celui-ci. Avec Trident 21.07, ce que l'utilisateur demande est l'espace inscriptible et Trident définit le nombre `snapshotReserve` comme le pourcentage du volume entier. Cela ne s'applique pas à `ontap-nas-economy`. Voir l'exemple suivant pour comprendre 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 la demande de l'utilisateur dans la demande PVC. La commande `volume show` devrait afficher des résultats similaires à cet exemple :

Les backends existants issus d'installations précédentes provisionneront les volumes comme expliqué ci-dessus lors de la mise à niveau de Trident. Pour les volumes que vous avez créés avant la mise à niveau, vous devez redimensionner leurs volumes pour que la modification soit prise en compte. Par exemple, un PVC de 2 Gio avec `snapshotReserve=50` aboutissait auparavant à un volume offrant 1 Gio d'espace accessible en écriture. En redimensionnant le volume à 3 Gio, par exemple, l'application disposera de 3 Gio d'espace accessible en écriture sur un volume de 6 Gio.

Exemples de configuration minimale

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.

REMARQUE	Si vous utilisez Amazon FSx sur NetApp ONTAP avec Trident, la recommandation est de spécifier les noms DNS des LIF au lieu des 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 backend pour éviter d'avoir à mettre à jour manuellement la définition du backend après le basculement et le retour en arrière pendant "Réplication et récupération de SVM".

Pour une transition et un retour en arrière sans heurt, spécifiez le SVM en utilisant `managementLIF` et omettez les `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 par certificat

Voici un exemple de configuration minimale du backend. `clientCertificate`, `clientPrivateKey`, et `trustedCACertificate` (facultatif, si vous utilisez une autorité de certification de confiance) sont renseignés dans `backend.json` et prennent respectivement les valeurs encodées en base64 du certificat client, de la clé privée et du certificat de l'autorité de certification de confiance.

```
---
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 règles d'export automatique

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

```
---
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` l'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'utilisation d'Amazon FSx for ONTAP avec des volumes SMB

Le `smbShare`paramètre est requis 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 du backend avec nameTemplate

```
---
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 backends avec pools virtuels

Dans les exemples de fichiers de définition de backend ci-dessous, des valeurs par défaut spécifiques sont définies pour tous les pools de stockage, comme `spaceReserve` à aucun, `spaceAllocation` à faux, et `encryption` à faux. Les pools virtuels sont définis dans la section storage.

Trident définit les étiquettes de provisionnement dans le champ « Commentaires ». Les commentaires sont définis sur FlexVol pour `ontap-nas` ou sur 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 simplicité, les administrateurs de stockage peuvent définir des étiquettes par pool virtuel et regrouper les volumes par étiquette.

Dans ces exemples, certains pools de stockage définissent leurs propres `spaceReserve`, `spaceAllocation`, et `encryption` valeurs, et certains pools remplacent les valeurs par défaut.

Exemple 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 NAS ONTAP FlexGroup

```
---
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"
```

Map backends vers StorageClasses

Les définitions de StorageClass suivantes font référence à Exemples de backends avec pools virtuels. En utilisant le champ `parameters.selector`, chaque StorageClass indique quels pools virtuels peuvent être utilisés pour héberger un volume. Le volume possédera les aspects définis dans le pool virtuel choisi.

- Le `protection-gold` StorageClass sera associé au premier et au deuxième pool virtuel dans le `ontap-nas-flexgroup` backend. Ce sont les seuls pools offrant une protection de niveau or.

```
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 associé au troisième et au quatrième pool virtuel dans le `ontap-nas-flexgroup` backend. 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 correspondra au quatrième pool virtuel dans le `ontap-nas` backend. Il s'agit du seul pool offrant une configuration de pool de stockage pour les applications 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 correspondra au troisième pool virtuel dans le `ontap-nas` backend et au deuxième pool virtuel dans le `ontap-nas-economy` backend. Ce sont les seuls pools proposant 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 déterminera quel pool virtuel est sélectionné et s'assurera que l'exigence de stockage est respectée.

Mise à jour `dataLIF` après la configuration initiale

Vous pouvez modifier le dataLIF après la configuration initiale en exécutant la commande suivante pour fournir le nouveau fichier JSON backend avec le dataLIF mis à jour.

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

REMARQUE	Si des PVC sont connectés à un ou plusieurs pods, vous devez mettre hors service tous les pods correspondants, puis les remettre en service afin que la nouvelle dataLIF prenne effet.

Exemples de SMB sécurisés

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 SMB sécurisé. Le 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 for NetApp ONTAP

Utilisez Trident avec Amazon FSx for NetApp ONTAP

"Amazon FSx for 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.

Exigences

En plus de "Exigences de 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.
- Un système de fichiers Amazon FSx for NetApp ONTAP existant et une machine virtuelle de stockage (SVM) accessibles depuis les nœuds de travail de votre cluster.
- Nœuds de travail préparés pour "NFS ou iSCSI".

REMARQUE	Assurez-vous de suivre les étapes de préparation des nœuds requises pour Amazon Linux et Ubuntu "Amazon Machine Images" (AMIs) en fonction de votre type d'AMI EKS.

Considérations

- Volumes SMB :
  - Les volumes SMB sont pris en charge à l'aide du pilote `ontap-nas` uniquement.
  - Les volumes SMB ne sont pas pris en charge avec l'add-on Trident EKS.
  - Trident prend uniquement en charge les volumes SMB montés sur des pods exécutés sur des nœuds Windows. Consultez "Préparez-vous à provisionner des volumes SMB" pour plus de détails.
- Avant Trident 24.02, les volumes créés sur les systèmes de fichiers Amazon FSx avec les sauvegardes automatiques activées ne pouvaient pas être supprimés par Trident. Pour éviter ce problème dans Trident 24.02 ou version ultérieure, spécifiez le `fsxFilesystemID`, AWS `apiRegion`, AWS `apikey`, et AWS `secretKey` dans le fichier de configuration du backend pour AWS FSx for ONTAP.

REMARQUE	Si vous spécifiez un rôle IAM à Trident, vous pouvez omettre de spécifier les champs `apiRegion`, `apiKey`, et `secretKey` à Trident explicitement. Pour plus d'informations, consultez "Options de configuration et exemples pour FSx for ONTAP".

Utilisation simultanée de Trident SAN/iSCSI et du pilote 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 multipath requise sur les nœuds peut entrer en conflit avec le pilote CSI Amazon Elastic Block Store (EBS). Pour garantir que le multipath fonctionne sans interférer avec les disques EBS sur le même nœud, vous devez exclure EBS de votre configuration multipath. Cet exemple montre un fichier `multipath.conf` qui inclut les paramètres Trident requis tout en excluant les disques EBS du multipath :

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

Authentification

Trident propose deux modes d'authentification.

- Authentification par identifiants (recommandée) : stocke les identifiants en toute sécurité dans AWS Secrets Manager. Vous pouvez utiliser l `fsxadmin` utilisateur de 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 portant un nom différent mais ayant le même rôle. Amazon FSx for NetApp ONTAP dispose d'un `fsxadmin` utilisateur qui est un remplacement limité de l'utilisateur du cluster ONTAP `admin`. Nous recommandons fortement d'utiliser `vsadmin` avec Trident.

- Basé sur un certificat : Trident communiquera avec la SVM de 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 l'authentification pour votre type de pilote :

- "Authentification NAS ONTAP"
- "Authentification SAN ONTAP"

Amazon Machine Images (AMIs) testées

Le cluster EKS prend en charge différents systèmes d'exploitation, mais AWS a optimisé certaines images machine Amazon (AMIs) pour les conteneurs et EKS. Les AMIs suivantes ont été testées avec NetApp Trident 25.02.

AMI	NAS	NAS-economy	iSCSI	iSCSI-economy
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 l'AMI que vous recherchez ne figure pas dans cette liste, cela ne signifie pas qu'elle n'est pas prise en charge ; cela signifie simplement qu'elle n'a pas été testée. Cette liste sert de guide pour les AMI dont le fonctionnement est avéré.

Tests effectués avec:

- EKS version : 1.32
- Méthode d'installation : Helm 25.06 et en tant qu'AWS add-On 25.06
- Pour NAS, les protocoles NFSv3 et NFSv4.1 ont été testés.
- Pour le SAN, seul l'iSCSI a été testé, pas le NVMe-oF.

Tests effectués :

- Créer : Storage Class, pvc, pod
- Supprimer : pod, pvc (standard, qtree/lun – économique, NAS avec sauvegarde AWS)

Pour plus d'informations

- "Documentation Amazon FSx for NetApp ONTAP"
- "Article de blog sur Amazon FSx for NetApp ONTAP"

Créez un rôle IAM et un secret AWS

Vous pouvez configurer les pods Kubernetes pour accéder aux ressources AWS en s'authentifiant en tant que rôle AWS IAM au lieu de fournir des identifiants AWS explicites.

REMARQUE	Pour vous authentifier à l'aide d'un rôle AWS IAM, vous devez disposer d'un cluster Kubernetes déployé à l'aide d'EKS.

Créer un secret AWS Secrets Manager

Étant donné que Trident utilisera des API sur un vserver FSx pour gérer le stockage pour vous, il aura besoin d'identifiants pour le faire. La méthode sécurisée pour transmettre ces identifiants consiste à utiliser un secret AWS Secrets Manager. Par conséquent, si vous n'en possédez pas déjà un, vous devrez créer un secret AWS Secrets Manager contenant les identifiants 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 stratégie IAM

Trident a également besoin des autorisations AWS pour fonctionner correctement. Par conséquent, vous devez créer une stratégie qui donne à Trident les autorisations dont il a besoin.

Les exemples suivants créent une stratégie IAM à l'aide de l'AWS CLI :

```
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 de JSON de stratégie :

```
{
  "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 du compte de service (IRSA)

Vous pouvez configurer un compte de service Kubernetes pour qu'il assume un rôle AWS Identity and Access Management (IAM) avec EKS Pod Identity ou IAM role for Service account association (IRSA). Tous les Pods configurés pour utiliser le compte de service peuvent alors accéder à tout service AWS auquel le rôle a des autorisations d'accès.

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 en utilisant la commande AWS CLI suivante :

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

Pour plus d'informations, consultez "Configurer l'agent d'identité du pod Amazon EKS".

Créer trust-relationship.json:

Créez trust-relationship.json pour permettre au Service Principal EKS d'assumer ce rôle pour l'identité du pod. Créez ensuite un rôle avec cette stratégie d'approbation :

```
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"
      ]
    }
  ]
}
```

Associez la stratégie de rôle au rôle IAM:

Associez la stratégie 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 compte de service (IRSA)

Utilisation de l'interface de ligne de commande AWS :

```
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 fichier `trust-relationship.json` :

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

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

Associez le rôle IAM à la stratégie IAM:

Une fois le rôle créé, associez la stratégie (qui a été créée à l'étape ci-dessus) au rôle à l'aide de cette commande :

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

Vérifiez que le fournisseur OICD est associé:

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

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

Si le résultat est vide, utilisez 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
```

Installer 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 :

- Helm
- Module complémentaire EKS

Si vous souhaitez utiliser la fonctionnalité de snapshot, installez l'add-on CSI snapshot controller. Consultez "Activer la fonctionnalité de snapshot pour les volumes CSI" pour plus d'informations.

Installez Trident via helm

Identité du pod

1. Ajoutez le dépôt 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 la commande `helm list` pour consulter les détails de l'installation tels que le nom, l'espace de noms, le chart, 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 compte de service (IRSA)

1. Ajoutez le dépôt Trident Helm :

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

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

```
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 la commande `helm list` pour consulter les détails de l'installation tels que le nom, l'espace de noms, le chart, 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 que iSCSI est activé sur votre machine cliente. Si vous utilisez AL2023 Worker node OS, vous pouvez automatiser l'installation du client iSCSI en ajoutant le paramètre node prep lors de l'installation avec 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

L'add-on Trident EKS inclut les derniers correctifs de sécurité, les corrections de bogues et est validé par AWS pour fonctionner avec Amazon EKS. L'add-on EKS vous permet de garantir de manière cohérente que vos clusters Amazon EKS sont sécurisés et stables, et de réduire la quantité de travail nécessaire pour installer, configurer et mettre à jour les add-ons.

Prérequis

Assurez-vous de disposer des éléments suivants avant de configurer le module complémentaire Trident pour AWS EKS :

- Un compte de cluster Amazon EKS avec abonnement complémentaire
- Autorisations AWS pour la AWS marketplace :
  `"aws-marketplace:ViewSubscriptions",`
  `"aws-marketplace:Subscribe",`
  `"aws-marketplace:Unsubscribe`
- Type d'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 for NetApp ONTAP existant

Activez le module complémentaire Trident pour AWS

Console de gestion

1. Ouvrez la console Amazon EKS à 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 **Modules complémentaires** puis sélectionnez **Obtenir plus de modules complémentaires**.
5. Suivez ces étapes pour sélectionner le module complémentaire :
   a. Faites défiler vers le bas jusqu'à la section **AWS Marketplace add-ons** et tapez **"Trident"** dans la zone de recherche.
   b. Cochez la case à cocher située dans le coin supérieur droit de la boîte Trident by NetApp.
   c. Sélectionnez **Next**.
6. Sur la page des paramètres **Configure selected add-ons**, procédez comme suit :

REMARQUE	Ignorez ces étapes si vous utilisez l'association Pod Identity.

   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 optionnels :
      - 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 role-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 Override comme méthode de résolution des conflits, un ou plusieurs des paramètres de l'add-on existant peuvent être remplacés par les paramètres de l'add-on 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 résultant pour résoudre le conflit. Avant de sélectionner cette option, assurez-vous que l'add-on Amazon EKS ne gère pas des paramètres que vous devez gérer vous-même.

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

   Une fois l'installation du logiciel complémentaire terminée, vous voyez votre logiciel complémentaire installé.

AWS CLI

1. Créez le `add-on.json` fichier :

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 Remplacez `<role ARN>` par l'ARN du rôle qui a été créé à l'étape précédente.

2. Installez le logiciel complémentaire Trident EKS.

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

eksctl

La commande suivante permet d'installer le module complémentaire Trident EKS :

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

Mettre 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 logiciel complémentaire NetApp Trident CSI.
4. Sélectionnez l'onglet **Add-ons**.
5. Sélectionnez **Trident par NetApp** puis sélectionnez **Modifier**.
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 optionnels** et modifiez-les si nécessaire.
   c. Sélectionnez **Save changes**.

AWS CLI

L'exemple suivant met à jour le add-on 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 add-on Trident CSI FSxN. 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 logiciel complémentaire avec la version renvoyée sous UPDATE AVAILABLE dans le résultat 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 ; vous recevez un message d'erreur pour vous aider à résoudre le conflit. Avant de spécifier cette option, assurez-vous que le module complémentaire Amazon EKS ne gère pas de paramètres que vous devez gérer, car ces paramètres sont écrasés avec cette option. Pour plus d'informations sur les autres options de ce paramètre, consultez "Modules complémentaires". Pour plus d'informations sur la gestion des champs Kubernetes d'Amazon EKS, consultez "Gestion des champs Kubernetes".

Désinstallez/supprimez le module complémentaire Trident EKS

Vous avez deux options pour supprimer un add-on Amazon EKS :

- **Conserver le logiciel complémentaire sur votre cluster** – Cette option supprime la gestion de tous les paramètres par Amazon EKS. Elle supprime également la capacité d'Amazon EKS à vous notifier des mises à jour et à mettre à jour automatiquement le logiciel complémentaire Amazon EKS après que vous ayez initié une mise à jour. Cependant, elle conserve le logiciel complémentaire sur votre cluster. Cette option fait du logiciel complémentaire une installation autogérée, plutôt qu'un logiciel complémentaire Amazon EKS. Avec cette option, il n'y a pas d'interruption pour le logiciel complémentaire. Conservez l' `--preserve`option dans la commande pour préserver le logiciel complémentaire.
- **Supprimez entièrement le logiciel complémentaire de votre cluster** – NetApp recommande de supprimer le module complémentaire Amazon EKS de votre cluster uniquement s'il n'y a aucune ressource sur votre cluster qui en dépend. Supprimez l'option `--preserve` du `delete`commande pour supprimer le logiciel complémentaire.

REMARQUE	Si le logiciel complémentaire a un compte IAM associé, le compte IAM n'est pas supprimé.

Console de gestion

1. Ouvrez la console Amazon EKS à 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 logiciel complémentaire Trident CSI NetApp.
4. Sélectionnez l'onglet **Add-ons** puis sélectionnez **Trident par NetApp**.*
5. Sélectionnez **Remove**.
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 logiciel complémentaire, sélectionnez **Conserver sur le cluster**. Procédez ainsi si vous souhaitez conserver le logiciel complémentaire sur votre cluster afin de pouvoir gérer vous-même tous les paramètres du logiciel complémentaire.
   b. Saisissez **netapp_trident-operator**.
   c. Sélectionnez **Remove**.

AWS CLI

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 Kubernetes StorageClass" 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 Créer un StorageClass Kubernetes sans configuration automatique du backend 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 devrait avoir découvert les pools sur le backend.

```
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 for ONTAP

Vous pouvez intégrer Trident à 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 de détails sur le conducteur, consultez "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 une configuration backend Trident (TBC) à partir d'un 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 backend 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 Bottlerocket, ajoutez les `mountOptions` requis 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 des volumes d'un type donné.

Attribut	Type	Valeurs	Offre	Demande	Soutenu par
médias[1]	chaîne	hdd, hybride, ssd	Le pool contient des médias de ce type ; hybride signifie les deux	Type de média spécifié	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, solidfire-san
provisioningType	chaîne	mince, épais	Pool prend en charge cette méthode de provisionnement	Méthode de provisionnement spécifiée	épais : all ontap ; mince : all ontap & solidfire-san
backendType	chaîne	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, solidfire-san, azure-netapp-files, ontap-san-economy	Pool appartient à ce type de backend	Backend spécifié	Tous les drivers
instantanés	bool	vrai, faux	Pool prend en charge les volumes avec snapshots	Volume avec instantanés activés	ontap-nas, ontap-san, solidfire-san

Attribut	Type	Valeurs	Offre	Demande	Soutenu par
clones	bool	vrai, faux	Pool prend en charge le clonage des volumes	Volume avec clones activés	ontap-nas, ontap-san, solidfire-san
chiffrement	bool	vrai, faux	Pool prend en charge les volumes chiffrés	Volume avec chiffrement activé	ontap-nas, ontap-nas-economy, ontap-nas-flexgroups, ontap-san
Op E/S par sec	int	entier positif	Pool est capable de garantir des IOPS dans cette plage	Volume garanti ces IOPS	solidfire-san

[1] : Non pris en charge par ONTAP Select ou les systèmes FSx for ONTAP

Consultez "Objets Kubernetes et Trident" pour plus de détails sur la manière dont les classes de stockage interagissent avec le `PersistentVolumeClaim` et les paramètres permettant de contrôler la façon 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, utilisez donc `kubectl` pour le 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 devrait avoir découvert les pools sur le backend.

```
kubectl get sc basic-csi
```

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

Provisionner des volumes SMB

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

Configuration avancée du backend et exemples

Consultez le tableau suivant pour les options de configuration du backend :

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 le stockage backend	Nom du pilote + "_" + dataLIF
`managementLIF`	Adresse IP d'un cluster ou d'une LIF de gestion SVM. Un nom de domaine complet (FQDN) peut être spécifié. Il est possible de configurer l'utilisation d'adresses IPv6 si Trident a été installé avec l'option IPv6. Les adresses IPv6 doivent être définies entre crochets, par exemple : [28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]. Si vous fournissez le `fsxFilesystemID` sous le champ `aws`, il n'est pas nécessaire de fournir le `managementLIF` car Trident récupère les informations SVM `managementLIF` depuis AWS. Vous devez donc fournir les identifiants d'un utilisateur sous la SVM (par exemple : vsadmin) et l'utilisateur doit avoir le rôle `vsadmin`.	"10.0.0.1", "[2001:1234:abcd::fefe]"

Paramètre	Description	Exemple
`dataLIF`	Adresse IP de l'interface logique de protocole (LIF). **Pilotes ONTAP NAS** : 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 complet (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 la configuration initiale. **Pilotes ONTAP SAN** : ne pas spécifier pour iSCSI. Trident utilise ONTAP Selective LUN Map pour découvrir les LIF iSCSI nécessaires à l'établissement d'une session multipath. Un avertissement est généré si dataLIF est explicitement défini. Il est possible de configurer l'utilisation d'adresses IPv6 si Trident a été installé avec l'option 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'export [Booléen]. En utilisant les options `autoExportPolicy` et `autoExportCIDRs`, Trident peut gérer les règles d'export automatiquement.	`false`
`autoExportCIDRs`	Liste des CIDR pour filtrer les adresses IP des nœuds Kubernetes lorsque `autoExportPolicy` est activé. En utilisant les options `autoExportPolicy` et `autoExportCIDRs`, Trident peut gérer les règles d'export automatiquement.	"["0.0.0.0/0", "::/0"]"
`labels`	Ensemble d'étiquettes arbitraires au format JSON à appliquer aux volumes	""

Paramètre	Description	Exemple
`clientCertificate`	Valeur encodée en Base64 du certificat client. Utilisé pour l'authentification par certificat	""
`clientPrivateKey`	Valeur encodée en Base64 de la clé privée du client. Utilisé pour l'authentification par certificat	""
`trustedCACertificate`	Valeur encodée en Base64 du certificat d'autorité de certification de confiance. Facultatif. Utilisé pour l'authentification par certificat.	""
`username`	Nom d'utilisateur pour se connecter au cluster ou à la SVM. Utilisé pour l'authentification par identifiants. Par exemple, vsadmin.	
`password`	Mot de passe permettant de se connecter au cluster ou à la SVM. Utilisé pour l'authentification par identifiants.	
`svm`	Machine virtuelle de stockage à utiliser	Dérivé si un LIF de gestion SVM est spécifié.
`storagePrefix`	Préfixe utilisé lors du provisionnement de nouveaux volumes dans la SVM. Ne peut pas être modifié après création. Pour mettre à jour ce paramètre, vous devrez créer un nouveau backend.	`trident`
`limitAggregateUsage`	**Ne pas spécifier pour Amazon FSx for NetApp ONTAP.** Les `fsxadmin` et `vsadmin` fournis ne contiennent pas les autorisations requises pour récupérer l'utilisation agrégée et la limiter à l'aide de Trident.	Ne pas utiliser.
`limitVolumeSize`	L'approvisionnement échoue si la taille du volume demandée dépasse cette valeur. Limite également la taille maximale des volumes qu'il gère pour les qtrees et les LUN, et l' `qtreesPerFlexvol`option permet de personnaliser le nombre maximal de qtrees par volume FlexVol	"" (non appliqué par défaut)
`lunsPerFlexvol`	Nombre maximal de LUN par volume FlexVol, doit être compris entre 50 et 200. SAN uniquement.	"100"

Paramètre	Description	Exemple
`debugTraceFlags`	Options de débogage à utiliser lors du dépannage. Exemple, {"api":false, "method":true} Ne pas utiliser `debugTraceFlags` sauf si vous effectuez un dépannage et avez besoin d'un journal détaillé.	null
`nfsMountOptions`	Liste d'options de montage NFS séparées par des virgules. Les options de montage pour les volumes persistants Kubernetes sont généralement spécifiées dans les classes de stockage, mais si aucune option de montage n'est spécifiée dans une classe de stockage, Trident utilisera les options de montage spécifiées dans le fichier de configuration du backend de stockage. Si aucune option de montage n'est spécifiée ni dans la classe de stockage ni dans le fichier de configuration, Trident n'appliquera aucune option de montage au volume persistant associé.	""
`nasType`	Configurez la création de volumes NFS ou SMB. Les options sont `nfs`, `smb` ou null. **Doit être défini sur `smb` pour les volumes SMB.** La valeur null correspond par défaut à des volumes NFS.	`nfs`
`qtreesPerFlexvol`	Nombre maximal de Qtrees par volume FlexVol, doit être compris dans la plage [50, 300]	`"200"`
`smbShare`	Vous pouvez spécifier l'un des éléments suivants : 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 les backends Amazon FSx for ONTAP.	`smb-share`

Paramètre	Description	Exemple
`useREST`	Paramètre booléen à utiliser pour les API REST ONTAP. Lorsqu'il est défini sur `true`, Trident utilisera les API REST ONTAP pour communiquer avec le backend. Cette fonctionnalité nécessite ONTAP 9.11.1 ou une version ultérieure. De plus, le rôle de connexion ONTAP utilisé doit avoir accès à l'application `ontap`. Cela est satisfait par les rôles prédéfinis `vsadmin` et `cluster-admin`.	`false`
`aws`	Vous pouvez spécifier les éléments suivants dans le fichier de configuration pour AWS FSx for 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 la SVM FSx à stocker dans AWS Secrets Manager. - `name` : Nom de ressource Amazon (ARN) du secret, qui contient les informations d'identification de la SVM. - `type` : Définissez sur `awsarn`. Consultez "Créer un secret AWS Secrets Manager" pour plus d'informations.	

Options de configuration backend pour le provisionnement des volumes

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

Paramètre	Description	Défaut
`spaceAllocation`	Allocation d'espace pour les LUNs	`true`
`spaceReserve`	Mode de réservation d'espace ; "none" (fin) ou "volume" (épais)	`none`
`snapshotPolicy`	Stratégie de snapshot à utiliser	`none`

Paramètre	Description	Défaut
`qosPolicy`	Groupe de règles QoS à attribuer aux volumes créés. Choisissez l'un des qosPolicy ou adaptiveQosPolicy par pool de stockage ou backend. L'utilisation des groupes de règles QoS avec Trident requiert 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 individuellement à chaque composant. Un groupe de règles QoS partagé impose une limite au débit total de toutes les charges de travail.	""
`adaptiveQosPolicy`	Groupe de règles QoS adaptatives à attribuer aux volumes créés. Choisissez l'un des qosPolicy ou adaptiveQosPolicy par pool de stockage ou backend. Non pris en charge par ontap-nas-economy.	""
`snapshotReserve`	Pourcentage du volume réservé aux 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`. NVE doit être sous licence et activé sur le cluster pour utiliser cette option. Si NAE est activé sur le backend, tout volume provisionné dans Trident sera activé pour NAE. Pour plus d'informations, consultez : "Comment Trident fonctionne avec NVE et NAE".	`false`
`luksEncryption`	Activez le chiffrement LUKS. Consultez "Utilisez Linux Unified Key Setup (LUKS)". SAN uniquement.	""
`tieringPolicy`	Politique de hiérarchisation à utiliser `none`	
`unixPermissions`	Mode pour les nouveaux volumes. **Laisser vide pour les volumes SMB.**	""

Paramètre	Description	Défaut
`securityStyle`	Style de sécurité pour les nouveaux volumes. NFS prend en charge `mixed` et `unix` styles de sécurité. SMB prend en charge `mixed` et `ntfs` styles de sécurité.	La valeur par défaut de NFS est `unix`. La valeur par défaut de SMB 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 le 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 un PVC de base avec un accès RWX qui est associé à un StorageClass nommé `basic-csi`.

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

Exemple d'utilisation de PVC avec iSCSI

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

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

Créer un PVC

Étapes

1. Créez le PVC.

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

2. Vérifiez l'état du PVC.

```
kubectl get pvc
```

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

Consultez "Objets Kubernetes et Trident" pour plus de détails sur la manière dont les classes de stockage interagissent avec le `PersistentVolumeClaim` et les paramètres permettant de contrôler la façon dont Trident provisionne les volumes.

Déployer une application

Une fois la classe de stockage et le PVC créés, vous pouvez monter le PV sur un pod. Cette section présente la commande et la configuration d'exemple pour associer le PV à un pod.

Déployer une application exemple

Étapes

1. Montez le volume dans un pod.

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

Ces exemples montrent des 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 en utilisant `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
```

Configurez le module complémentaire Trident EKS sur un cluster EKS

NetApp 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. L'add-on NetApp Trident EKS inclut les derniers

correctifs de sécurité, les corrections de bogues, et est validé par AWS pour fonctionner avec Amazon EKS. L'add-on EKS vous permet de garantir de manière cohérente que vos clusters Amazon EKS sont sécurisés et stables, et de réduire la quantité de travail nécessaire pour installer, configurer et mettre à jour les add-ons.

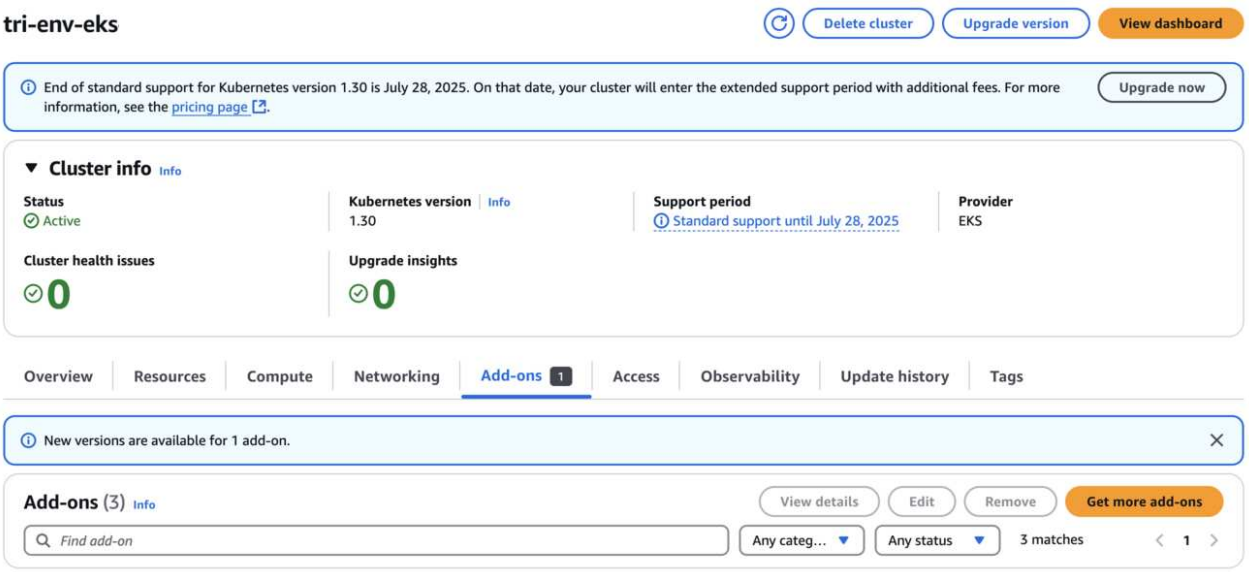
Prérequis

Assurez-vous de disposer des éléments suivants avant de configurer le module complémentaire Trident pour AWS EKS :

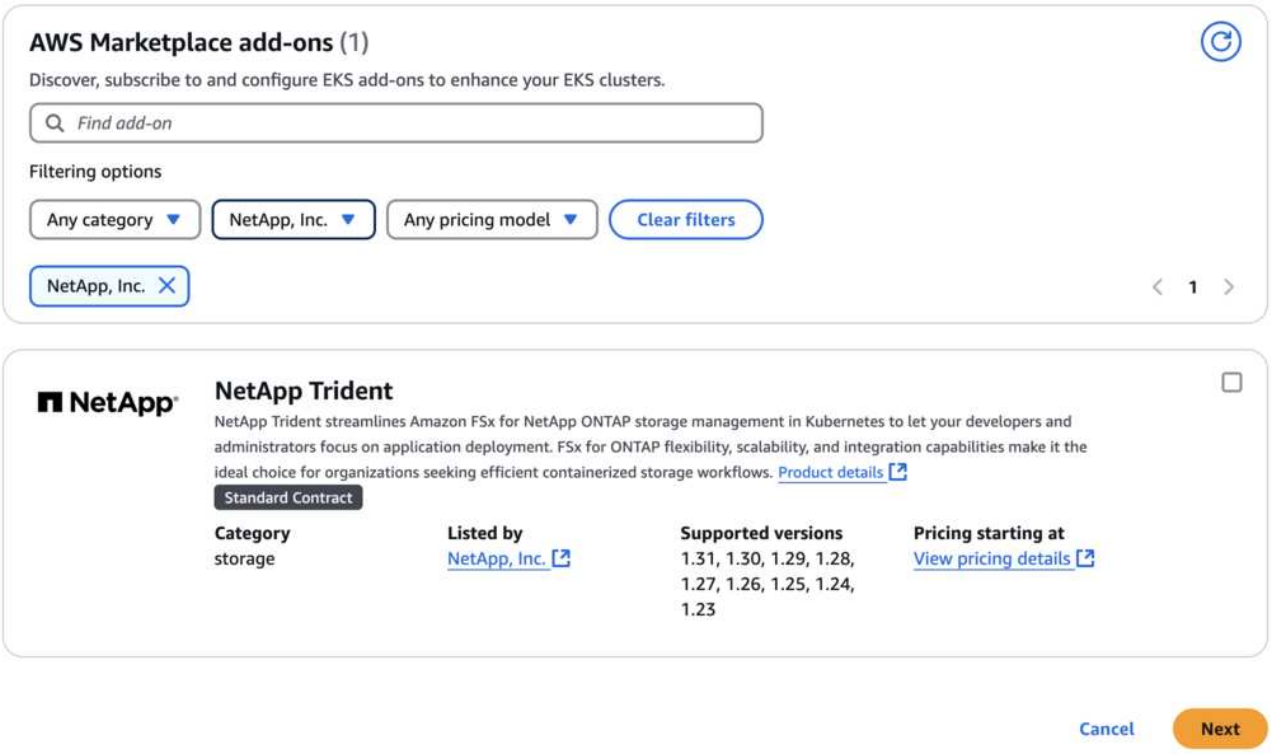
- Un compte de cluster Amazon EKS disposant des autorisations nécessaires pour gérer les modules complémentaires. Consultez "Modules complémentaires Amazon EKS".
- Autorisations AWS pour la AWS marketplace :
  `"aws-marketplace:ViewSubscriptions",`
  `"aws-marketplace:Subscribe",`
  `"aws-marketplace:Unsubscribe`
- Type d'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 for NetApp ONTAP existant

Étapes

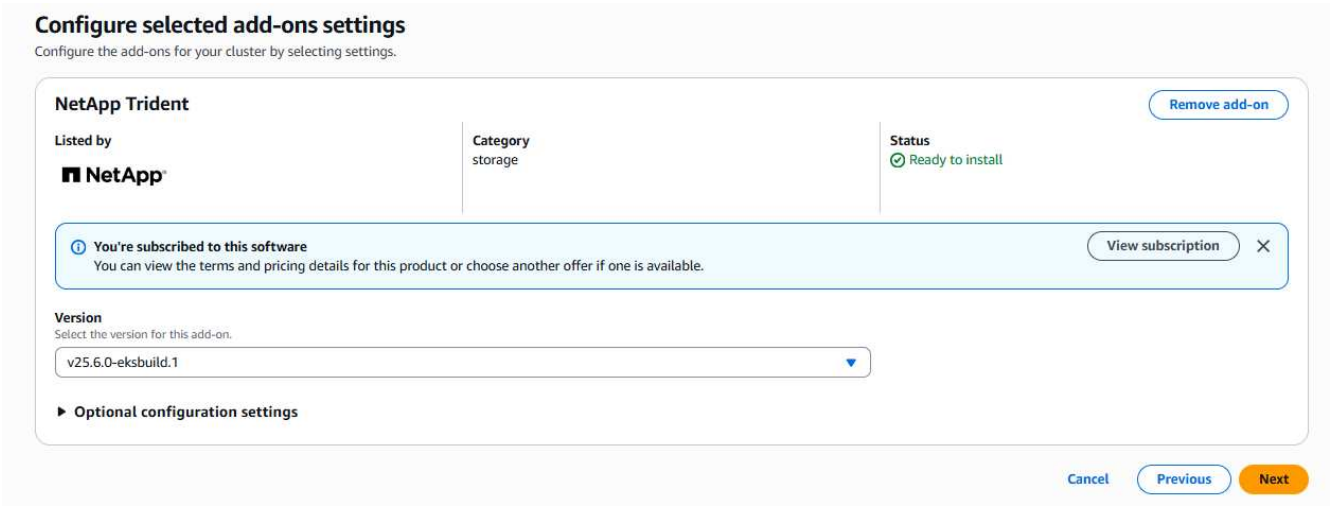
1. Veillez à créer un rôle IAM et un secret AWS pour permettre aux pods EKS d'accéder aux ressources AWS. Pour obtenir des instructions, consultez "Créez un rôle IAM et un secret AWS".
2. Sur votre cluster Kubernetes EKS, accédez à l'onglet **Add-ons**.



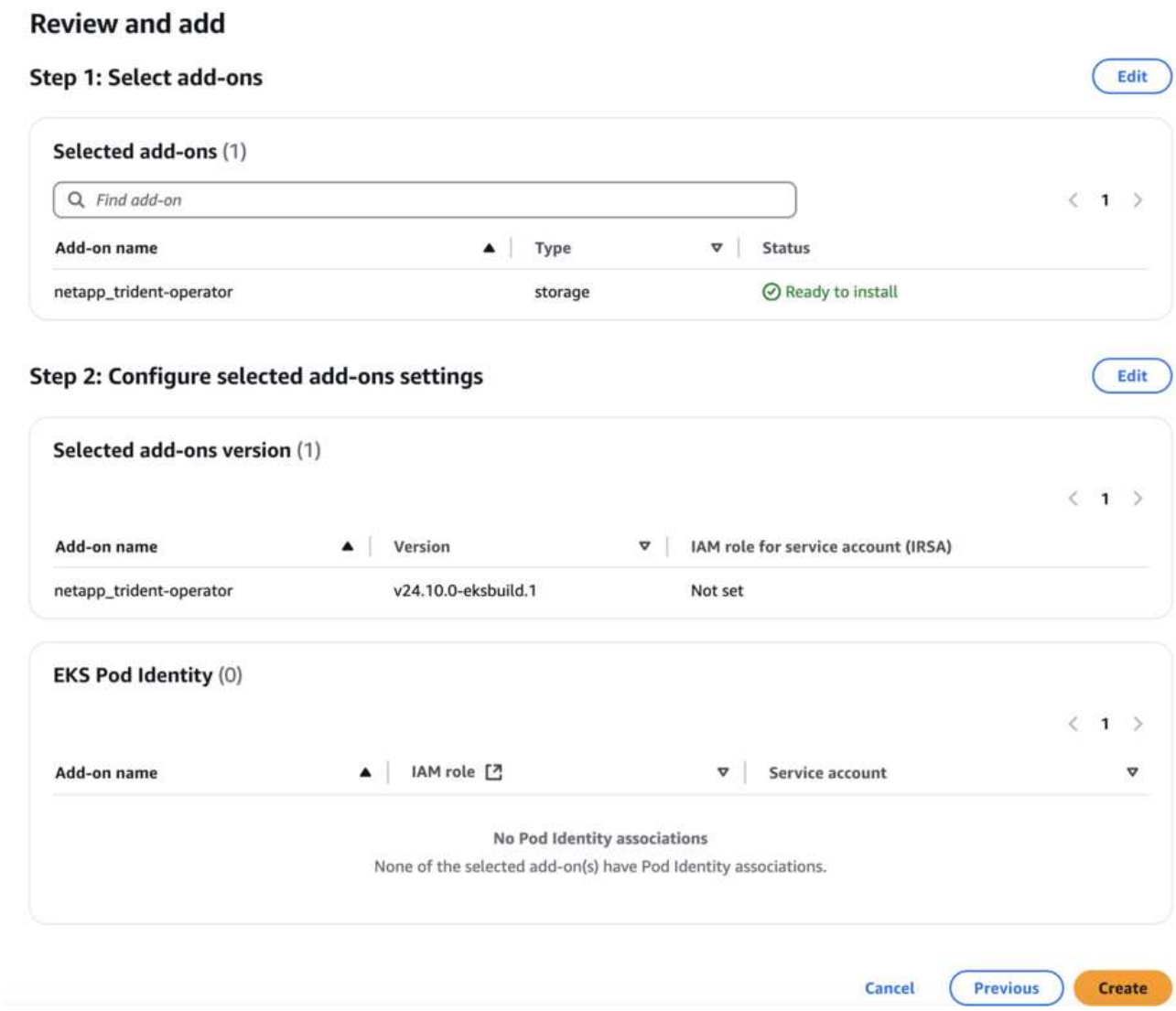
3. Accédez à **AWS Marketplace add-ons** et choisissez la catégorie *storage*.



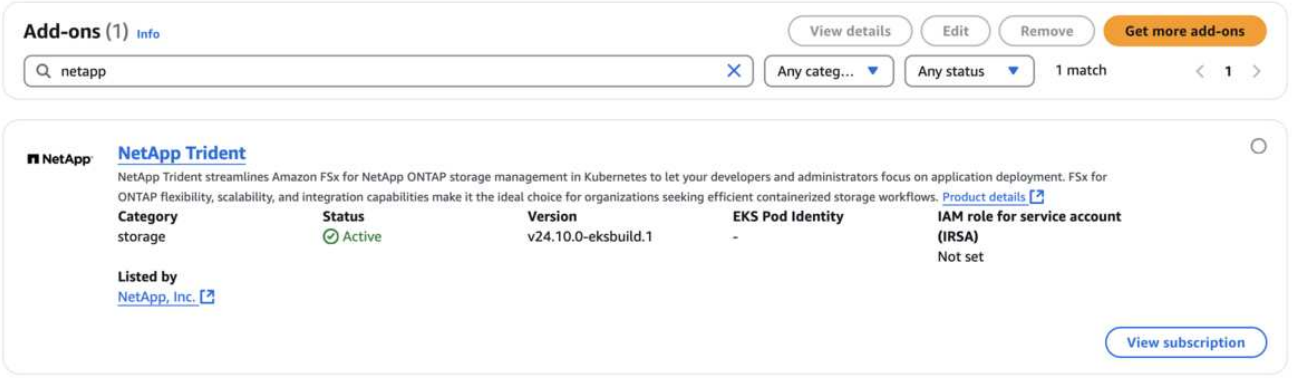
4. Repérez **NetApp Trident** et cochez la case de l'extension Trident, puis cliquez sur **Suivant**.
5. Choisissez la version souhaitée du module complémentaire.



6. Configurez les paramètres du module complémentaire requis.



7. Si vous utilisez IRSA (IAM roles for service account), reportez-vous aux étapes de configuration supplémentaires "ici".
8. Sélectionnez **Create**.
9. Vérifiez que le statut du module complémentaire 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 "Configurer le backend de stockage".

Installez/désinstallez l'add-on Trident EKS à l'aide de la CLI

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

La commande suivante 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 Trident EKS add-on 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
```

Créer des backends avec kubectl

Un backend définit la relation entre Trident et un système de stockage. Il indique à Trident comment communiquer avec ce système de stockage et comment Trident doit provisionner des volumes à partir de celui-ci. Après l'installation de Trident, l'étape suivante consiste à créer un backend. La `TridentBackendConfig` Custom Resource Definition (CRD) permet de créer et de gérer des backends Trident directement via l'interface Kubernetes. Vous pouvez le faire en utilisant `kubectl` ou l'outil CLI équivalent pour votre distribution Kubernetes.

`TridentBackendConfig`

`TridentBackendConfig` (`tbc`, `tbconfig`, `tbackendconfig`) est une CRD frontale avec espace de noms qui vous permet de gérer les backends Trident à l'aide de `kubectl`. Les administrateurs Kubernetes et de stockage peuvent désormais créer et gérer des backends directement via l'interface de ligne de commande Kubernetes sans avoir besoin d'un utilitaire en ligne de commande dédié (`tridentctl`).

Lors de la création d'un `TridentBackendConfig` objet, les événements suivants se produisent :

- Un backend est créé automatiquement par Trident en fonction de la configuration que vous fournissez. Ceci est représenté en interne par une `TridentBackend` (`tbe`, `tridentbackend`) CR.
- Le `TridentBackendConfig` est lié de manière unique à un `TridentBackend` qui a été créé par

Trident.

Chacune `TridentBackendConfig` maintient une correspondance un-à-un avec une `TridentBackend`. La première est l'interface fournie à l'utilisateur pour concevoir et configurer les backends ; la seconde est la façon dont Trident représente l'objet backend réel.

AVERTISSEMENT	`TridentBackend` Les CR sont créés automatiquement par Trident. Vous **ne devez pas** les modifier. Si vous souhaitez mettre à jour les backends, faites-le en modifiant l'objet `TridentBackendConfig`.

Voir l'exemple suivant pour 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 dans le répertoire "trident-installer" pour des configurations types pour la plateforme/le service de stockage souhaité.

Le `spec` prend des paramètres de configuration spécifiques au backend. Dans cet exemple, le backend utilise le `ontap-san` storage driver et utilise les paramètres de configuration qui sont présentés ici. Pour la liste des options de configuration pour votre storage driver souhaité, consultez le "Informations de configuration du backend pour votre pilote de stockage".

La `spec` section inclut également `credentials` et `deletionPolicy` des champs, qui sont nouvellement introduits dans le `TridentBackendConfig` CR :

- `credentials`: Ce paramètre est obligatoire et contient les identifiants utilisés pour l'authentification auprès du système/service de stockage. Ceci est défini sur un Secret Kubernetes créé par l'utilisateur. Les identifiants ne peuvent pas être transmis en clair et cela entraînera une erreur.
- `deletionPolicy` : Ce champ définit ce qui doit se produire lorsque le `TridentBackendConfig` est supprimé. Il peut prendre l'une des deux valeurs possibles :
  - `delete`: Cela entraîne la suppression de la `TridentBackendConfig` CR et du backend associé. Il s'agit de la valeur par défaut.
  - `retain` : Lorsqu'un `TridentBackendConfig` CR est supprimé, la définition du backend reste présente et peut être gérée avec `tridentctl`. La configuration de la politique de suppression sur `retain` permet aux utilisateurs de revenir à une version antérieure (pré-21.04) et de conserver les backends créés. La valeur de ce champ peut être mise à jour après qu'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 backend est défini sur le nom de l'objet `TridentBackendConfig` (metadata.name). Il est recommandé de définir explicitement les noms des backends à l'aide de `spec.backendName`.
ASTUCE	Les backends qui ont été créés avec `tridentctl` n'ont pas d'objet `TridentBackendConfig` associé. Vous pouvez choisir de gérer ces backends avec `kubectl` en créant un CR `TridentBackendConfig`. Il faut veiller à spécifier des paramètres de configuration identiques (tels que `spec.backendName`, `spec.storagePrefix`, `spec.storageDriverName`, etc.). Trident associera automatiquement le nouvel objet `TridentBackendConfig` au backend préexistant.

Aperçu des étapes

Pour créer un nouveau backend en utilisant `kubectl`, vous devez procéder comme suit :

1. Créez un "Secret Kubernetes". Le secret contient les identifiants dont Trident a besoin pour communiquer avec le cluster/service de stockage.
2. Créez un `TridentBackendConfig` objet. Celui-ci contient des informations spécifiques sur le cluster/service de stockage et fait référence au secret créé à l'étape précédente.

Après avoir créé un backend, vous pouvez observer son statut en utilisant `kubectl get tbc <tbc-name> -n <trident-namespace>` et recueillir des informations supplémentaires.

Étape 1 : Créer un secret Kubernetes

Créez un secret contenant les identifiants d'accès pour le backend. Ceci 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 qui doivent figurer dans le Secret pour chaque plateforme de stockage :

Description des champs secrets de la plateforme de stockage	Secret	Description des champs
Azure NetApp Files	clientID	L'identifiant client issu de l'enregistrement d'une application
Element (NetApp HCI/SolidFire)	Point de terminaison	MVIP pour le cluster SolidFire avec les identifiants du locataire
ONTAP	nom d'utilisateur	Nom d'utilisateur pour se connecter au cluster/SVM. Utilisé pour l'authentification basée sur les identifiants
ONTAP	mot de passe	Mot de passe pour se connecter au cluster/SVM. Utilisé pour l'authentification basée sur les identifiants
ONTAP	clientPrivateKey	Valeur encodée en Base64 de la clé privée du client. Utilisée pour l'authentification basée sur un certificat
ONTAP	chapUsername	Nom d'utilisateur entrant. Requis si useCHAP=true. Pour `ontap-san` et `ontap-san-economy`
ONTAP	chapInitiatorSecret	Secret de l'initiateur CHAP. Requis si useCHAP=true. Pour `ontap-san` et `ontap-san-economy`
ONTAP	chapTargetUsername	Nom d'utilisateur cible. Requis si useCHAP=true. Pour `ontap-san` et `ontap-san-economy`
ONTAP	chapTargetInitiatorSecret	Secret de l'initiateur de la cible CHAP. Requis si useCHAP=true. Pour `ontap-san` et `ontap-san-economy`

Le secret créé à cette étape sera référencé dans le champ `spec.credentials` de l'objet `TridentBackendConfig` qui est créé à l'étape suivante.

Étape 2 : Créer la `TridentBackendConfig` demande de changement

Vous êtes maintenant prêt à créer votre `TridentBackendConfig` CR. Dans cet exemple, un backend qui utilise le `ontap-san` driver est créé à l'aide de l' `TridentBackendConfig` objet présenté 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 : Vérifiez le statut 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 backend a été créé avec succès et lié au `TridentBackendConfig` CR.

La phase peut prendre l'une des valeurs suivantes :

- `Bound` : Le `TridentBackendConfig`CR est associé à un backend, et ce backend contient `configRef`défini sur l' `TridentBackendConfig`uid du CR.
- `Unbound` : Représenté à l'aide de `""`. L'objet `TridentBackendConfig` n'est pas lié à un backend. Tous les nouveaux CR créés `TridentBackendConfig` sont par défaut dans cette phase. Après le changement de phase, il ne peut plus revenir à l'état Unbound.
- `Deleting` : Le `TridentBackendConfig`CR `deletionPolicy` a été configuré pour être supprimé. Lorsque le `TridentBackendConfig`CR est supprimé, il passe à l'état Deleting.
  - Si aucune revendication de volume persistant (PVC) n'existe sur le backend, la suppression du `TridentBackendConfig` entraînera Trident à supprimer le backend ainsi que le `TridentBackendConfig` CR.
  - Si une ou plusieurs PVC sont présentes sur le backend, il passe en état de suppression. Le `TridentBackendConfig` CR entre ensuite également en phase de suppression. Le backend et

`TridentBackendConfig` ne sont supprimés qu'après la suppression de toutes les PVC.

- `Lost` : Le backend associé au `TridentBackendConfig` CR a été supprimé accidentellement ou intentionnellement et le `TridentBackendConfig` CR possède toujours une référence vers le backend supprimé. Le `TridentBackendConfig` CR peut toujours être supprimé, quelle que soit la valeur de `deletionPolicy`.
- `Unknown` : Trident ne parvient pas à déterminer l'état ou l'existence du backend associé au `TridentBackendConfig` CR. Par exemple, si le serveur API ne répond pas ou si le `tridentbackends.trident.netapp.io` CRD est manquant. Cela peut nécessiter une intervention.

À ce stade, le backend est créé avec succès ! Plusieurs opérations supplémentaires peuvent être gérées, telles que "mises à jour du backend et suppressions du backend".

(Facultatif) Étape 4 : Obtenir plus de détails

Vous pouvez exécuter la commande suivante pour obtenir plus d'informations sur votre backend :

```
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
```

De plus, vous pouvez également obtenir un dump 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 l' `backendName` et le `backendUUID` du backend qui a été créé en réponse à la `TridentBackendConfig` CR. Le `lastOperationStatus` champ représente le statut de la dernière opération de la `TridentBackendConfig` CR, qui peut être déclenchée par l'utilisateur (par exemple, si l'utilisateur a modifié quelque chose dans `spec`) ou déclenchée par Trident (par exemple, lors des redémarrages de Trident). Il peut être soit Success soit Failed. `phase` représente le statut de la relation entre la `TridentBackendConfig` CR et le backend. Dans l'exemple ci-dessus, `phase` a la valeur Bound, ce qui signifie que la `TridentBackendConfig` CR est associée au backend.

Vous pouvez exécuter la commande `kubectl -n trident describe tbc <tbc-cr-name>` pour obtenir les détails des journaux d'événements.

AVERTISSEMENT	Vous ne pouvez pas mettre à jour ou supprimer un backend qui contient un objet associé `TridentBackendConfig` en utilisant `tridentctl`. Pour comprendre les étapes nécessaires pour passer de `tridentctl` à `TridentBackendConfig`, "voir ici".

Gérer les backends

Effectuez la gestion du backend avec kubectl

Découvrez comment effectuer des opérations de gestion backend en utilisant `kubectl`.

Supprimer un backend

En supprimant un `TridentBackendConfig`, vous indiquez à Trident de supprimer ou de conserver les backends (selon `deletionPolicy`). Pour supprimer un backend, assurez-vous que `deletionPolicy` est défini sur delete. Pour supprimer uniquement le `TridentBackendConfig`, assurez-vous que `deletionPolicy` est défini sur retain. Cela garantit que le backend 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 qui étaient utilisés par `TridentBackendConfig`. L'utilisateur Kubernetes est responsable du nettoyage des secrets. Il faut faire attention lors de la suppression des secrets. Vous ne devez supprimer les secrets que s'ils ne sont pas utilisés par les backends.

Afficher les backends 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 backends existants. Cette liste inclura également les backends qui ont été créés avec `tridentctl`.

Mettre à jour un backend

Il peut exister plusieurs raisons de mettre à jour un backend :

- Les identifiants d'accès au système de stockage ont changé. Pour mettre à jour les identifiants, le Secret Kubernetes utilisé dans l' `TridentBackendConfig`objet doit être mis à jour. Trident mettra automatiquement à jour le backend avec les identifiants les plus récents fournis. Exécutez la commande suivante pour mettre à jour le Secret Kubernetes :

  ```
  kubectl apply -f <updated-secret-file.yaml> -n trident
  ```

- Les paramètres (tels que le nom de la ONTAP SVM utilisée) doivent être mis à jour.
  - Vous pouvez mettre à jour `TridentBackendConfig` les objets directement via Kubernetes à l'aide de la commande suivante :

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

- Vous pouvez également apporter des modifications à la `TridentBackendConfig`CR existante 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 backend, le backend conserve sa dernière configuration connue. Vous pouvez consulter les journaux pour en déterminer la cause en exécutant `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 de mise à jour.

Effectuez la gestion du backend avec tridentctl

Découvrez comment effectuer des opérations de gestion backend en utilisant `tridentctl`.

Créer un backend

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

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

Si la création du backend échoue, quelque chose n'allait pas avec la configuration du backend. Vous pouvez consulter les journaux pour en déterminer la cause en exécutant la commande suivante :

```
tridentctl logs -n trident
```

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

Supprimer un backend

Pour supprimer un backend de Trident, procédez comme suit :

1. Récupérer le nom du backend :

```
tridentctl get backend -n trident
```

2. Supprimez le backend :

```
tridentctl delete backend <backend-name> -n trident
```

REMARQUE	Si Trident a provisionné des volumes et des instantanés à partir de ce backend qui existent encore, la suppression du backend empêche le provisionnement de nouveaux volumes par celui-ci. Le backend continuera d'exister dans un état « Deleting ».

Afficher les backends existants

Pour afficher les backends que Trident connaît, procédez comme suit :

- Pour obtenir un résumé, 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 backend

Après avoir créé un nouveau fichier de configuration, exécutez la commande suivante :

```
tridentctl update backend <backend-name> -f <backend-file> -n trident
```

Si la mise à jour du backend échoue, cela signifie qu'il y a un problème avec la configuration du backend ou que vous avez tenté une mise à jour invalide. Vous pouvez consulter les journaux pour en déterminer la cause en exécutant la commande suivante :

```
tridentctl logs -n trident
```

Après avoir identifié et corrigé le problème avec le fichier de configuration, vous pouvez simplement exécuter la commande `update` à nouveau.

Identifiez les classes de stockage qui utilisent un backend

Voici un exemple du type de questions auxquelles vous pouvez répondre avec le JSON que `tridentctl` génère pour les objets backend. Cela utilise l' `jq` utilitaire, que vous devez installer.

```
tridentctl get backend -o json | jq '[.items[] | {backend: .name,
storageClasses: [.storage[].storageClasses]|unique}]'
```

Cela s'applique également aux backends qui ont été créés en utilisant `TridentBackendConfig`.

Passer d'une option de gestion du backend à une autre

Découvrez les différentes manières de gérer les backends dans Trident.

Options pour la gestion des backends

Avec l'introduction de `TridentBackendConfig`, les administrateurs disposent désormais de deux méthodes distinctes pour gérer les backends. Cela soulève les questions suivantes :

- Les backends créés à l'aide de `tridentctl` peuvent-ils être gérés avec `TridentBackendConfig` ?
- Les backends créés à l'aide de `TridentBackendConfig` peuvent-ils être gérés à l'aide de `tridentctl` ?

Gérer `tridentctl` les backends à l'aide de `TridentBackendConfig`

Cette section couvre les étapes nécessaires pour gérer les backends qui ont été créés à l'aide de `tridentctl` directement via l'interface Kubernetes en créant des `TridentBackendConfig` objets.

Cela s'appliquera aux scénarios suivants :

- Les backends préexistants, qui n'ont pas de `TridentBackendConfig` parce qu'ils ont été créés avec `tridentctl`.
- De nouveaux backends ont été créés avec `tridentctl`, tandis que d'autres `TridentBackendConfig` objets existent.

Dans les deux cas, les backends resteront présents, Trident planifiant les volumes et opérant dessus. Les administrateurs ont alors deux choix :

- Continuez à utiliser `tridentctl` pour gérer les backends qui ont été créés avec.
- Liez les backends créés à l'aide de `tridentctl` à un nouvel objet `TridentBackendConfig`. Cela signifie que les backends seront gérés à l'aide de `kubectl` et non `tridentctl`.

Pour gérer un backend préexistant à l'aide de `kubectl`, vous devrez créer un `TridentBackendConfig` qui se lie au backend existant. Voici un aperçu de la façon dont cela fonctionne :

1. Créez un secret Kubernetes. Le secret contient les informations d'identification dont Trident a besoin pour communiquer avec le cluster/service de stockage.
2. Créez un `TridentBackendConfig` objet. Celui-ci contient des informations spécifiques sur le cluster/service de stockage et fait référence au secret créé à l'étape précédente. Veillez à spécifier des paramètres de configuration identiques (tels que `spec.backendName`, `spec.storagePrefix`, `spec.storageDriverName`, etc.). `spec.backendName` doit être défini sur le nom du backend existant.

Étape 0 : Identifier le backend

Pour créer une `TridentBackendConfig` qui se lie à un backend existant, vous devrez obtenir la configuration du backend. Dans cet exemple, supposons qu'un backend ait 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 : Créer un secret Kubernetes

Créez un Secret qui contient les identifiants pour le backend, 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 liera automatiquement à la `ontap-nas-backend` préexistante (comme dans cet exemple). Assurez-vous que les conditions suivantes sont remplies :

- Le même nom de backend est défini dans `spec.backendName`.
- Les paramètres de configuration sont identiques à ceux du backend d'origine.
- Les pools virtuels (le cas échéant) doivent conserver le même ordre que dans le backend d'origine.
- Les informations d'identification sont fournies via un Secret Kubernetes et non en texte clair.

Dans ce cas, le `TridentBackendConfig` ressemblera à ceci :

```
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 : Vérifiez le statut du `TridentBackendConfig` CR

Après la `TridentBackendConfig` création, sa phase doit être `Bound`. Il doit également refléter le même nom de backend et le même UUID que le backend 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 backend sera désormais entièrement géré à l'aide de l' `tbc-ontap-nas-backend` `TridentBackendConfig`objet.

Gérer `TridentBackendConfig` les backends à l'aide de `tridentctl`

```
`tridentctl` peut être utilisé pour lister les backends qui ont été créés
à l'aide de  `TridentBackendConfig`. De plus, les administrateurs peuvent
également choisir de gérer entièrement ces backends via  `tridentctl` en
supprimant  `TridentBackendConfig` et en s'assurant que
`spec.deletionPolicy` est défini sur  `retain`.
```

Étape 0 : Identifier le backend

Par exemple, supposons que le backend suivant ait 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 |
+-------------------+----------------
+--------------------------------------+--------+---------+
```

D'après le résultat, on constate que `TridentBackendConfig` a été créé avec succès et est lié à un backend [observer l'UUID du backend].

Étape 1 : Confirmer `deletionPolicy`est défini sur `retain`

Examinons la valeur de `deletionPolicy`. Celle-ci doit être définie sur `retain`. Cela garantit que lorsqu'une `TridentBackendConfig` CR est supprimée, la définition backend sera toujours présente et pourra 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 passez pas à l'étape suivante à moins que `deletionPolicy` soit défini sur `retain`.

Étape 2 : Supprimer le `TridentBackendConfig` CR

La dernière étape consiste à supprimer le `TridentBackendConfig` CR. Après avoir confirmé que le `deletionPolicy` est défini sur `retain`, vous pouvez procéder à 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 l `TridentBackendConfig`objet, Trident le supprime simplement sans supprimer le backend lui-même.

Créer et gérer des classes de stockage

Créer une classe de stockage

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

Configurer un objet StorageClass Kubernetes

Le "Objet Kubernetes StorageClass" identifie Trident comme le provisionneur 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
```

Consultez "Objets Kubernetes et Trident" pour plus de détails sur la manière dont les classes de stockage interagissent avec le `PersistentVolumeClaim` et les paramètres permettant de contrôler la façon dont Trident provisionne les volumes.

Créer une classe de stockage

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

Étapes

1. Il s'agit d'un objet Kubernetes, utilisez donc `kubectl` pour le 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 devrait avoir découvert les pools sur le backend.

```
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"
        ]
      }
    }
  ]
}
```

Exemples de classe de stockage

Trident fournit "Définitions simples de classes de stockage pour des backends spécifiques".

Vous pouvez également modifier `sample-input/storage-class-csi.yaml.templ` fichier fourni avec le programme d'installation et remplacer *`BACKEND_TYPE`* par 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 consulter les classes de stockage existantes, définir une classe de stockage par défaut, identifier le backend de la classe de stockage et supprimer des 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 d'une 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 les détails 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 introduit la possibilité de définir une classe de stockage par défaut. Il s'agit de la classe de stockage qui sera utilisée pour provisionner un volume persistant si un utilisateur n'en spécifie pas dans une revendication de volume persistant (PVC).

- Définissez une classe de stockage par défaut en définissant l'annotation `storageclass.kubernetes.io/is-default-class` sur true dans la définition de la classe de stockage. Conformément à la spécification, toute autre valeur ou l'absence de l'annotation est interprétée comme false.
- 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 en utilisant 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 d'installation de Trident qui incluent cette annotation.

REMARQUE	Il ne devrait y avoir qu'une seule classe de stockage par défaut dans votre cluster à la fois. Kubernetes ne vous empêche pas techniquement d'en avoir plusieurs, mais il se comportera comme s'il n'y avait aucune classe de stockage par défaut.

Identifier le backend d'une classe de stockage

Voici un exemple du type de questions auxquelles vous pouvez répondre avec le JSON que `tridentctl` génère pour les objets backend Trident. Cela utilise l' `jq` utilitaire, que vous devrez peut-être installer d'abord.

```
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>` devrait être remplacé par votre classe de stockage.

Tous les volumes persistants qui ont été créés via cette classe de stockage resteront intacts, et Trident continuera de les gérer.

REMARQUE	Trident impose un espace vide `fsType` pour les volumes qu'il crée. Pour les backends iSCSI, il est recommandé de l'imposer `parameters.fsType` dans le StorageClass. Vous devez supprimer les StorageClasses existants et les recréer avec `parameters.fsType` spécifié.

Provisionner et gérer les volumes

Provisionner un volume

Créez un PersistentVolumeClaim (PVC) qui utilise le StorageClass Kubernetes configuré pour demander l'accès au PV. Vous pouvez ensuite monter le PV sur un pod.

Aperçu

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 PVC, vous pouvez monter le volume dans un pod.

Créer le PVC

Étapes

1. Créez le PVC.

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

2. Vérifiez l'état du PVC.

```
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 en utilisant `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 PersistentVolumeClaim

Ces exemples montrent les options de configuration de base des PVC.

PVC avec accès RWO

Cet exemple montre un PVC de base avec un accès RWO qui est associé à un StorageClass nommé `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 montre un PVC de base pour NVMe/TCP avec accès RWO qui est associé à un StorageClass nommé `protection-gold`.

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

Exemples de manifestes Pod

Ces exemples montrent des configurations de base pour attacher le PVC 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
```

Consultez "Objets Kubernetes et Trident" pour plus de détails sur la manière dont les classes de stockage interagissent avec le `PersistentVolumeClaim` et les paramètres permettant de contrôler la façon dont Trident provisionne les volumes.

Étendre les volumes

Trident permet aux utilisateurs de Kubernetes d'étendre leurs volumes après leur création. Consultez les informations relatives aux configurations requises pour étendre les volumes iSCSI, NFS, SMB, NVMe/TCP et FC.

Étendre un volume iSCSI

Vous pouvez étendre un volume persistant iSCSI (PV) à l'aide du provisionneur CSI.

REMARQUE	L'extension de volume iSCSI est prise en charge par les `ontap-san`, `ontap-san-economy`, `solidfire-san` pilotes et nécessite Kubernetes 1.16 et versions ultérieures.

Étape 1 : Configurer le StorageClass pour prendre en charge l'extension de volume

Modifiez la définition StorageClass pour définir le champ `allowVolumeExpansion` sur `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 un StorageClass déjà existant, modifiez-le pour inclure le `allowVolumeExpansion` paramètre.

Étape 2 : Créez un PVC avec le StorageClass que vous avez créé

Modifiez la définition du PVC 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 (PV) et l'associe à cette Persistent Volume Claim (PVC).

```
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 : Définir un pod auquel se fixe le PVC

Fixez le PV à un pod pour qu'il soit redimensionné. Il existe deux scénarios lors du redimensionnement d'un PV iSCSI :

- Si le PV est attaché à un pod, Trident étend le volume sur le backend de stockage, rescanne le périphérique et redimensionne le système de fichiers.
- Lors d'une tentative de redimensionnement d'un PV non attaché, Trident étend le volume sur le stockage backend. Après que le PVC est lié à un pod, Trident rescanne le périphérique et redimensionne le système de fichiers. Kubernetes met ensuite à jour la taille du PVC après que l'opération d'extension a été effectuée avec succès.

Dans cet exemple, un pod est créé qui 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 : Développer le PV

Pour redimensionner le PV créé de 1Gi à 2Gi, modifiez la définition du PVC et mettez à jour le `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'expansion

Vous pouvez vérifier que l'extension a fonctionné correctement en contrôlant la taille du PVC, du 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    |
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
```

Étendre un volume FC

Vous pouvez étendre un volume persistant (PV) FC en utilisant le provisionneur CSI.

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

Étape 1 : Configurer le StorageClass pour prendre en charge l'extension de volume

Modifiez la définition StorageClass pour définir le champ `allowVolumeExpansion` sur `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 un StorageClass déjà existant, modifiez-le pour inclure le `allowVolumeExpansion` paramètre.

Étape 2 : Créez un PVC avec le StorageClass que vous avez créé

Modifiez la définition du PVC 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 (PV) et l'associe à cette Persistent Volume Claim (PVC).

```
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 : Définir un pod auquel se fixe le PVC

Fixez le PV à un pod pour qu'il soit redimensionné. Il existe deux scénarios lors du redimensionnement d'un PV FC :

- Si le PV est attaché à un pod, Trident étend le volume sur le backend de stockage, rescanne le périphérique et redimensionne le système de fichiers.
- Lors d'une tentative de redimensionnement d'un PV non attaché, Trident étend le volume sur le stockage backend. Après que le PVC est lié à un pod, Trident rescanne le périphérique et redimensionne le système de fichiers. Kubernetes met ensuite à jour la taille du PVC après que l'opération d'extension a été effectuée

avec succès.

Dans cet exemple, un pod est créé qui 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 : Développer le PV

Pour redimensionner le PV créé de 1Gi à 2Gi, modifiez la définition du PVC et mettez à jour le `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'expansion

Vous pouvez vérifier que l'extension a fonctionné correctement en contrôlant la taille du PVC, du 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    |
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
```

Étendre 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 le StorageClass pour prendre en charge l'extension de volume

Pour redimensionner un volume persistant NFS, l'administrateur doit d'abord configurer la classe de stockage pour autoriser l'extension de volume en définissant le `allowVolumeExpansion` champ sur `true` :

```
cat storageclass-ontapnas.yaml
```

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontapnas
provisioner: csi.trident.netapp.io
parameters:
  backendType: ontap-nas
allowVolumeExpansion: true
```

Si vous avez déjà créé une classe de stockage sans cette option, vous pouvez simplement modifier la classe

de stockage existante en utilisant `kubectl edit storageclass` pour autoriser l'extension de volume.

Étape 2 : Créez un PVC avec le StorageClass que vous avez créé

```
cat pvc-ontapnas.yaml
```

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: ontapnas20mb
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 20Mi
  storageClassName: ontapnas
```

Trident devrait créer un PV NFS de 20 MiB 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 : Développer le PV

Pour redimensionner le PV nouvellement créé de 20 Mio à 1 Gio, modifiez le PVC et définissez `spec.resources.requests.storage` sur 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'expansion

Vous pouvez vérifier que le redimensionnement a fonctionné correctement en contrôlant la taille du PVC, du 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 introduit la scalabilité des contrôleurs grâce à une meilleure gestion de la concurrence entre plusieurs pilotes de stockage. Les clients peuvent identifier quels pilotes Trident prennent en charge la scalabilité des contrôleurs à disponibilité générale et quels pilotes sont disponibles en avant-première technique dans Trident 26.02. Cela permet de prendre des décisions de déploiement éclairées et d'assurer une gestion appropriée des risques pour les environnements Kubernetes évolutifs.

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, ainsi que la gestion du backend.

Lorsque la mise à l'échelle du contrôleur est activée, 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.

Prise en charge de l'évolutivité du contrôleur

Trident prend en charge l'évolutivité du contrôleur avec différents niveaux de maturité en fonction du pilote de stockage.

Disponibilité générale

Les pilotes suivants prennent en charge la mise à l'échelle du contrôleur à la disponibilité générale dans Trident 26.02 :

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

REMARQUE	Les `google-cloud-netapp-volumes` et `google-cloud-netapp-volumes-san` pilotes sont différents. Seul `google-cloud-netapp-volumes` est pris en charge. N'utilisez pas `google-cloud-netapp-volumes-san` dans les configurations ou exemples backend.

Activer la scalabilité du contrôleur

La scalabilité du contrôleur est contrôlée par l'option de configuration `enableConcurrency`. Cette option doit être explicitement activée 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 ressource personnalisée `TridentOrchestrator`.

Nouvelle installation

Créez ou modifiez le `TridentOrchestrator` CR avec `enableConcurrency` défini sur `true` :

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

Appliquez le 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 le paramètre a bien été appliqué :

```
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 niveau en utilisant 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 existant basé sur `tridentctl`, désinstallez et réinstallez 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 fonctionne avec la concurrence activée en consultant les journaux du pod du contrôleur :

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

Vous devriez voir une entrée de journal indiquant que la concurrence est activée.

Aperçu technique

Les pilotes suivants prennent en charge la mise à l'échelle du contrôleur en tant qu'aperçu technique dans Trident 26.02 :

- `nas-eco`
- `san-eco`

Pour ces pilotes :

- La concurrence des contrôleurs est disponible pour l'évaluation et les tests
- Le comportement peut changer dans les prochaines versions
- L'utilisation en environnement de production n'est pas recommandée

Comportement de concurrence

Lorsque la mise à l'échelle du contrôleur est activée :

- Trident remplace le verrou global unique par un verrouillage fin, par ressource
- Les opérations qui modifient la même ressource sont sérialisées afin de garantir la cohérence des données
- Les opérations qui ne font que lire à partir d'une ressource peuvent s'exécuter 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 la même interface logique de gestion (LIF), ils partagent cette limite de 20 requêtes

Limitations et considérations connues

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

- La concurrence est gérée en interne par le contrôleur Trident
- Il n'y a pas de limites de concurrence configurables par l'utilisateur dans cette version
- Le débit global dépend de :
  - Le pilote de stockage utilisé
  - Réactivité du backend
  - Performance du serveur API Kubernetes
- Une forte concurrence peut augmenter la charge sur les systèmes de stockage backend

Avertissements et limitations

Les limitations suivantes s'appliquent dans Trident 26.02 :

- Le comportement de mise à l'échelle du contrôleur n'est pas identique pour tous les pilotes
- Les pilotes en préversion technique peuvent présenter :
  - Performances irrégulières sous forte charge
  - Changements de comportement entre les versions
- Le débogage des opérations simultanées peut être plus complexe en raison de l'exécution parallèle
- Les indicateurs et les journaux peuvent afficher des résultats d'opérations entrelacées

Recommandations

- Utilisez les pilotes à disponibilité générale (GA) pour les environnements de production nécessitant une évolutivité élevée

- Évaluer les pilotes en préversion technique dans des environnements hors production
- Surveillez les performances du backend et du contrôleur lors du fonctionnement à grande échelle
- Évitez de présumer 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.

Exigences

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)

Limitations

- 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	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	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

Après avoir créé des stratégies Autogrow, vous pouvez les consulter, les mettre à jour et les supprimer selon vos besoins. Vous pouvez également surveiller quels volumes 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 actuellement cette politique. Testez d'abord les modifications dans un environnement hors production lorsque cela est possible.

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 utilisant la politique adoptent de nouveaux paramètres lors de la prochaine évaluation de la croissance.
- **Aucun redémarrage du volume nécessaire :** 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'autogrow utilisent la protection du finaliseur pour empêcher toute suppression accidentelle pendant que des 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`.

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. Ceci est dû 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"`. Cela remplace toute politique au niveau de la 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. Aucune autre demande de redimensionnement n'est créée pour ce volume, même si l'utilisation continue d'augmenter au-delà de la `usedThreshold`.

Par exemple, avec cette politique, Trident cesse d'augmenter le volume une fois qu'il atteint 100 GiB :

```
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 utilisant 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 `Deleting` état et attend que tous les volumes soient supprimés 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 la suppression de tous les volumes, le finaliseur est libéré et la policy est supprimée.

La fonction d'extension automatique fonctionne-t-elle avec tous les backends ONTAP ?

La fonctionnalité d'extension automatique prend en charge les protocoles NFS, iSCSI, FCP et NVMe. Cependant, les volumes de blocs bruts NVMe nécessitent ONTAP 9.16.1 ou une version ultérieure.

Les volumes existants dans les déploiements brownfield peuvent nécessiter la migration de la publication des volumes avant que l'autogrow ne prenne effet. Vérifiez l'état de la migration en consultant 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.

Aperçu et considérations

Vous pouvez importer un volume dans Trident pour :

- Conteneurisez une application et réutilisez son ensemble de données existant
- Utilisez un clone d'un ensemble de données pour une application éphémère
- Reconstruire un cluster Kubernetes défaillant
- Migrer les données d'application lors d'une reprise après sinistre

Considérations

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

- Trident peut importer uniquement les volumes ONTAP de type RW (lecture-écriture). 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 suggérons d'importer les volumes sans connexion active. Pour importer un volume utilisé activement, clonez le volume, puis effectuez l'importation.

AVERTISSEMENT	Ceci est particulièrement important pour les volumes de blocs, car Kubernetes ignorerait la connexion précédente et pourrait facilement associer un volume actif à un pod. Cela peut entraîner une corruption des données.

- Bien que `StorageClass` doive être spécifié sur un PVC, Trident ne l'utilise pas lors de l'importation. Les classes de stockage sont utilisées lors de la création du volume pour sélectionner parmi les pools disponibles en fonction des caractéristiques de stockage. Parce que le volume existe déjà, aucune sélection de pool n'est requise lors de l'importation. Par conséquent, l'importation ne sera pas un échec même si le volume existe sur un backend 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 le PVC. Après que le volume a été importé par le pilote de stockage, le PV est créé avec un ClaimRef vers le PVC.
  - La politique de récupération est initialement définie sur `retain` dans le PV. Après que Kubernetes a correctement lié le PVC et le PV, la politique de récupération est mise à jour pour correspondre à la politique de récupération de la Storage Class.
  - Si la politique de récupération de la Storage Class est `delete`, le volume de stockage sera supprimé lorsque le PV sera supprimé.
- Par défaut, Trident gère le PVC et renomme le FlexVol volume et le LUN sur le backend. Vous pouvez passer l' `--no-manage`option pour importer un volume non géré et l' `--no-rename`option pour conserver le nom du volume.
  - `--no-manage`* - Si vous utilisez le `--no-manage` flag, Trident n'effectue aucune opération supplémentaire sur le PVC ou le PV pendant tout 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 de volume et le redimensionnement de volume sont également ignorées.

◦ `--no-rename`* - Si vous utilisez le `--no-rename` indicateur, 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 `ontap-nas`, `ontap-san` (y compris les systèmes ASA r2), et les pilotes `ontap-san-economy`.

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 aux PVC et PV et sert à la fois à indiquer que le volume a été importé et si les PVC et PV sont gérés. 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 la commande `tridentctl import` pour 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). L'argument `-f` est obligatoire 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 volumes suivants pour les pilotes pris en charge.

ONTAP NAS et ONTAP NAS FlexGroup

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

REMARQUE

- Trident ne prend pas en charge l'importation de volumes à l'aide du `ontap-nas-economy`driver.
- Les `ontap-nas` et `ontap-nas-flexgroup` pilotes n'autorisent pas les noms de volume en double.

Chaque volume créé avec le `ontap-nas` driver est un volume FlexVol sur le cluster ONTAP. L'importation de volumes FlexVol avec le `ontap-nas` driver fonctionne de la même manière. Un volume FlexVol qui existe déjà sur un cluster ONTAP peut être importé comme un `ontap-nas` PVC. De même, les volumes FlexGroup peuvent être importés comme des `ontap-nas-flexgroup` PVC.

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 backend 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` backend :

```
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>
```

ONTAP SAN

Trident prend en charge l'importation de volumes à l'aide des `ontap-san` (iSCSI, NVMe/TCP et FC) et `ontap-san-economy` pilotes.

Trident peut importer des volumes SAN ONTAP FlexVol qui contiennent un seul LUN. Cela est cohérent avec le `ontap-san` driver, qui crée un volume FlexVol pour chaque PVC et un LUN dans le volume FlexVol. Trident importe le volume FlexVol et l'associe à la définition du PVC. Trident peut importer des volumes `ontap-san-economy` qui contiennent plusieurs LUN.

Les exemples suivants montrent comment importer des volumes gérés et non gérés :

Volume géré

Pour les volumes gérés, Trident renomme le volume FlexVol au format `pvc-<uuid>` et le LUN à l'intérieur du volume FlexVol en `lun0`.

L'exemple suivant importe le `ontap-san-managed` volume FlexVol qui est présent sur le `ontap_san_default` backend :

```
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` backend :

```
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 vous avez des LUN mappés à des igroups qui partagent un IQN avec l'IQN d'un nœud Kubernetes, comme illustré dans l'exemple suivant, vous recevrez l'erreur : `LUN already mapped to initiator(s) in this group`. Vous devrez supprimer l'initiateur ou dissocier le 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
```

Élément

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

REMARQUE	Le pilote Element prend en charge les noms de volumes identiques. Cependant, Trident renvoie une erreur s'il existe des noms de volumes en double. Pour contourner ce problème, clonez le volume, attribuez un nom de volume unique et importez le volume cloné.

L'exemple suivant importe un `element-managed` volume sur le backend `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` driver.

REMARQUE	Pour importer un volume Azure NetApp Files, identifiez le volume par son chemin de volume. Le chemin de 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 de volume est `importvol1`.

L'exemple suivant importe un `azure-netapp-files` volume sur le backend `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` driver.

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

Personnalisez les noms et les étiquettes des volumes

Avec Trident, vous pouvez attribuer des noms et des étiquettes explicites aux volumes que vous créez. Cela vous aide à identifier et à associer facilement les volumes à leurs ressources Kubernetes (PVC) respectives. Vous pouvez également définir des modèles au niveau du backend pour créer des noms et des étiquettes de volumes personnalisés ; tous les volumes que vous créez, importez ou clonez respecteront ces modèles.

Avant de commencer

Prise en charge des noms et étiquettes de volume personnalisables :

- Opérations de création, d'importation et de clonage de volume.
- Dans le cas du `ontap-nas-economy`driver, seul le nom du volume Qtree est conforme au modèle de nom.
- Dans le cas du `ontap-san-economy`driver, seul le nom du LUN est conforme au modèle de nom.

Limitations

- 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` pilotes.
- Les noms de volumes personnalisés ne s'appliquent pas aux volumes existants.

Comportements clés des noms de volumes personnalisables

- En cas d'échec dû à une syntaxe incorrecte dans un modèle de nom, la création du backend échoue. Cependant, si l'application du modèle échoue, le volume sera nommé selon 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 issu de la configuration du backend. Toute valeur de préfixe souhaitée peut être ajoutée directement au modèle.

Exemples de configuration backend avec modèle de nom et labels

Des modèles de noms personnalisés peuvent être définis au niveau racine et/ou au niveau 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 considérer

1. Dans le cas des importations de volumes, les étiquettes ne sont mises à jour que si le volume existant possède des étiquettes dans un format spécifique. Par exemple : `{"provisioning":{"Cluster":"ClusterA", "PVC": "pvcname"}}`.
2. Dans le cas des importations de volumes gérés, le nom du volume suit le modèle de nom défini au niveau racine dans la définition du backend.
3. Trident ne prend pas en charge l'utilisation d'un opérateur de découpage avec le préfixe de stockage.
4. Si les modèles ne produisent pas de noms de volumes uniques, Trident ajoutera quelques caractères aléatoires pour créer des noms de volumes uniques.
5. Si le nom personnalisé d'un volume NAS economy dépasse 64 caractères, Trident nommera les volumes selon la convention de nommage existante. Pour tous les autres pilotes ONTAP, si le nom du volume dépasse la limite de caractères, le processus de création du volume échoue.

Partager un volume NFS entre espaces de noms

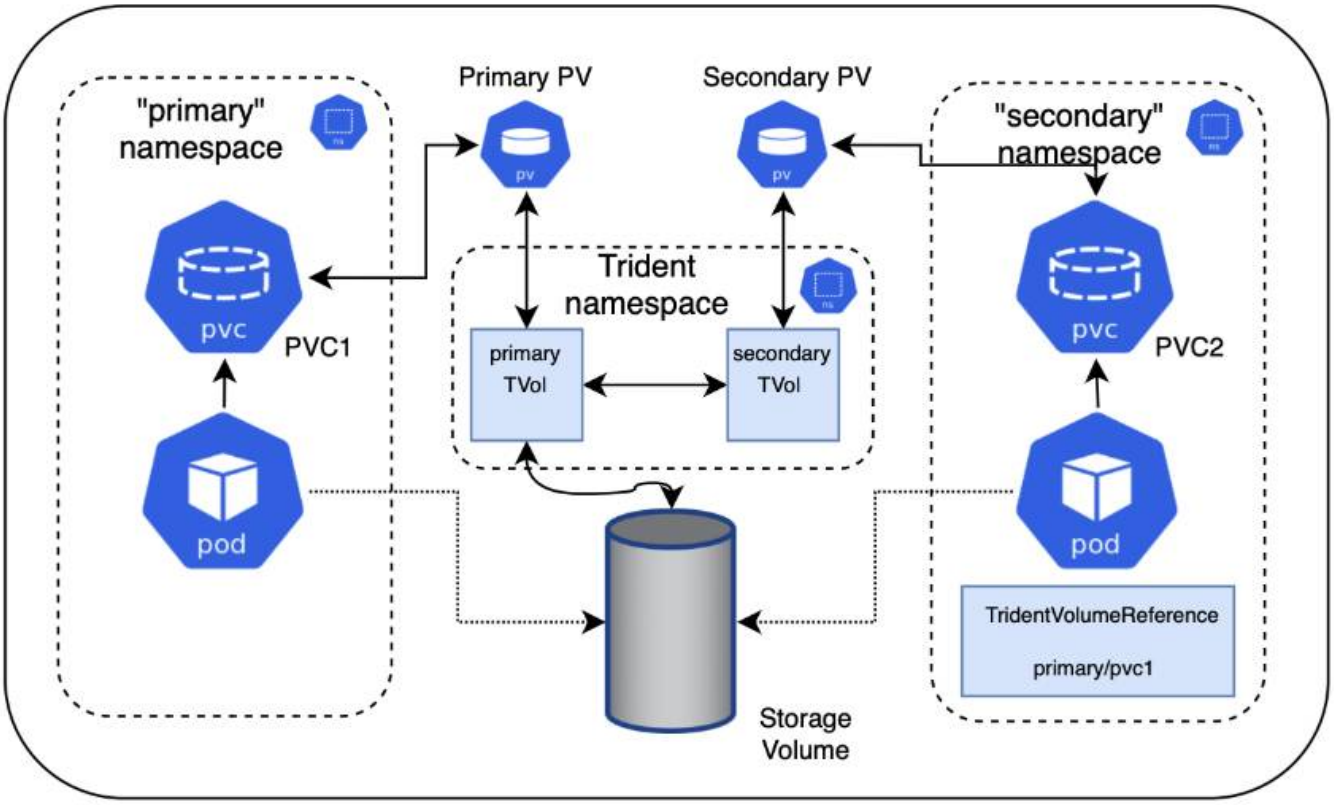
Avec Trident, vous pouvez créer un volume dans un espace de noms principal et le partager dans un ou plusieurs espaces de noms secondaires.

Caractéristiques

Le TridentVolumeReference CR vous permet de partager en toute sécurité des volumes NFS ReadWriteMany (RWX) entre un ou plusieurs espaces de noms Kubernetes. Cette solution native Kubernetes présente les avantages suivants :

- Plusieurs niveaux de contrôle d'accès pour garantir la sécurité
- Fonctionne avec tous les pilotes de volumes NFS Trident
- Aucune dépendance à tridentctl ni à toute autre fonctionnalité non native de Kubernetes

Ce diagramme illustre le partage de volumes NFS entre deux espaces de noms Kubernetes.



Démarrage rapide

Vous pouvez configurer le partage de volume NFS en quelques étapes seulement.

1 **Configurez le PVC source pour partager le volume**

Le propriétaire de l'espace de noms source accorde l'autorisation d'accéder aux données du PVC source.

2 **Autoriser la création d'un CR dans l'espace de noms de destination**

L'administrateur du cluster autorise le propriétaire de l'espace de noms de destination à créer la TridentVolumeReference CR.

3 **Créer TridentVolumeReference dans l'espace de noms de destination**

Le propriétaire de l'espace de noms de destination crée la TridentVolumeReference CR pour faire référence au PVC source.

4 **Créez le PVC subordonné dans l'espace de noms de destination**

Le propriétaire de l'espace de noms de destination crée le PVC subordonné pour utiliser la source de données du PVC source.

Configurez les espaces de noms source et de destination

Pour garantir la sécurité, le partage entre espaces de noms nécessite la collaboration et l'intervention 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 de l'utilisateur est défini à chaque étape.

Étapes

1. **Propriétaire de l'espace de noms source :** Créez le PVC (`pvc1` dans l'espace de noms source qui autorise le partage avec l'espace de noms de destination (`namespace2` en utilisant l'annotation `shareToNamespace`.

```
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 PV et son volume de stockage NFS backend.

   REMARQUE

   - Vous pouvez partager le PVC avec plusieurs espaces de noms à l'aide d'une liste séparée par des virgules. Par exemple, `trident.netapp.io/shareToNamespace: namespace2,namespace3,namespace4`.
   - Vous pouvez partager avec tous les espaces de noms en utilisant `*`. Par exemple, `trident.netapp.io/shareToNamespace: *`
   - Vous pouvez mettre à jour le PVC pour inclure l'annotation `shareToNamespace` à tout moment.

2. **Administrateur du cluster :** Assurez-vous qu'un contrôle d'accès basé sur les rôles (RBAC) approprié est en place pour accorder au propriétaire de l'espace de noms de destination l'autorisation de créer la TridentVolumeReference CR dans l'espace de noms de destination.
3. **Propriétaire de l'espace de noms de destination :** Créez une 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 (`pvc2` dans l'espace de noms de destination (`namespace2` en utilisant l'annotation `shareFromPVC` pour désigner le 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 celle du PVC source.

Résultats

Trident lit l' `shareFromPVC`annotation sur le PVC de destination et crée le PV de destination comme un volume subordonné sans ressource de stockage propre qui pointe vers le PV source et partage la ressource de stockage du PV source. Le PVC et le PV de destination apparaissent liés normalement.

Supprimer un volume partagé

Vous pouvez supprimer un volume partagé entre plusieurs espaces de noms. Trident supprimera l'accès au volume dans l'espace de noms source et maintiendra l'accès pour les autres espaces de noms qui partagent le volume. Lorsque tous les espaces de noms qui référencent le volume sont supprimés, Trident supprime le volume.

Utilisez `tridentctl get` pour interroger les volumes subordonnés

À l'aide de 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]
```

Drapeaux:

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

Limitations

- Trident ne peut pas empêcher les espaces de noms de destination d'écrire sur le volume partagé. Vous devez utiliser le verrouillage de fichiers ou d'autres processus pour éviter l'écrasement des données du volume partagé.
- Vous ne pouvez pas révoquer l'accès au PVC source en supprimant les `shareToNamespace` ou `shareFromNamespace` annotations ou en supprimant le `TridentVolumeReference` CR. Pour révoquer l'accès, vous devez supprimer le PVC subordonné.
- Les instantanés, les clones et la mise en miroir ne sont pas possibles sur les volumes subordonnés.

Pour plus d'informations

Pour en savoir plus sur l'accès aux volumes entre espaces de noms :

- Regardez la démo sur "NetAppTV".

Cloner des volumes entre espaces de noms

Avec Trident, vous pouvez créer de nouveaux volumes à l'aide de volumes existants ou de volumesnapshots provenant d'un espace de noms différent au sein du même cluster Kubernetes.

Prérequis

Avant de cloner des volumes, assurez-vous que les backends source et de destination sont du même type et ont la même classe de stockage.

REMARQUE	Le clonage entre espaces de noms est pris en charge uniquement pour les `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 volume en quelques étapes seulement.

1. **Configurez le PVC source pour cloner le volume**

Le propriétaire de l'espace de noms source accorde l'autorisation d'accéder aux données du PVC source.

2 Autoriser la création d'un CR dans l'espace de noms de destination

L'administrateur du cluster autorise le propriétaire de l'espace de noms de destination à créer la TridentVolumeReference CR.

3 Créer TridentVolumeReference dans l'espace de noms de destination

Le propriétaire de l'espace de noms de destination crée la TridentVolumeReference CR pour faire référence au PVC source.

4 Créez le PVC cloné dans l'espace de noms de destination

Le propriétaire de l'espace de noms de destination crée un PVC pour cloner le PVC de l'espace de noms source.

Configurez 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'intervention 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 de l'utilisateur est défini à chaque étape.

Étapes

1. **Propriétaire de l'espace de noms source :** Créez le PVC (`pvc1`) dans l'espace de noms source (`namespace1`) qui autorise le partage avec l'espace de noms de destination (`namespace2`) en utilisant l'annotation `cloneToNamespace`.

```
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 PV et son volume de stockage backend.

REMARQUE

- Vous pouvez partager le PVC avec plusieurs espaces de noms à l'aide d'une liste séparée par des virgules. Par exemple, `trident.netapp.io/cloneToNamespace: namespace2,namespace3,namespace4`.
- Vous pouvez partager avec tous les espaces de noms en utilisant `*`. Par exemple, `trident.netapp.io/cloneToNamespace: *`
- Vous pouvez mettre à jour le PVC pour inclure l'annotation `cloneToNamespace` 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 la TridentVolumeReference CR dans l'espace de noms de destination (`namespace2`).
3. **Propriétaire de l'espace de noms de destination :** Créez une 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 (`pvc2`) dans l'espace de noms de destination (`namespace2`) en utilisant les `cloneFromPVC` ou `cloneFromSnapshot`, et `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
```

Limitations

- Pour les PVC provisionnés à l'aide des pilotes ontap-nas-economy, les clones en lecture seule ne sont pas pris en charge.

Répliquer les volumes à l'aide de SnapMirror

Trident prend en charge les relations de miroir entre un volume source sur un cluster et le volume de destination sur le cluster homologue pour la réplication des données en vue de la reprise après sinistre. Vous pouvez utiliser une définition de ressource personnalisée (CRD) avec espace de noms, appelée Trident Mirror Relationship (TMR), pour effectuer les opérations suivantes :

- Créer des relations de miroir entre les volumes (PVCs)
- Supprimer les relations de miroir entre les volumes
- Briser les relations miroir
- Promouvoir le volume secondaire pendant des conditions de sinistre (basculements)
- Effectuer une transition sans perte des applications d'un cluster à l'autre (lors de basculements ou de migrations planifiés)

Prérequis de réplication

Assurez-vous que les conditions préalables suivantes sont remplies avant de commencer :

clusters ONTAP

- **Trident** : la version 22.10 ou ultérieure de Trident doit être présente sur les clusters Kubernetes source et de destination qui utilisent ONTAP comme backend.
- **Licences** : Les licences asynchrones ONTAP SnapMirror utilisant le Data Protection bundle doivent être activées sur les clusters ONTAP source et de destination. Consultez "SnapMirror aperçu des licences dans ONTAP" pour plus d'informations.

  À compter de ONTAP 9.10.1, toutes les licences sont fournies sous forme de fichier de licence NetApp (NLF), qui est un fichier unique permettant d'activer plusieurs fonctionnalités. Consultez "Licences incluses avec ONTAP One" pour plus d'informations.

REMARQUE	Seule la protection asynchrone SnapMirror est prise en charge.

Interconnexion

- **Cluster et SVM** : Les backends de stockage ONTAP doivent être appariés. Consultez "Aperçu du peering de cluster et de SVM" pour plus d'informations.

IMPORTANT	Assurez-vous que les noms SVM utilisés dans la relation de réplication entre deux clusters ONTAP sont uniques.

- **Trident et SVM** : les SVM distants appariés doivent être disponibles pour Trident sur le cluster de destination.

Pilotes pris en charge

NetApp Trident prend en charge la réplication de volumes avec la technologie NetApp SnapMirror en utilisant

des classes de stockage prises en charge par les pilotes suivants : **`ontap-nas`: NFS** `ontap-san`: iSCSI **`ontap-san`: FC** `ontap-san`: NVMe/TCP (nécessite la version ONTAP 9.15.1 au minimum)

REMARQUE	La réplication de volumes à l'aide de SnapMirror n'est pas prise en charge pour les systèmes ASA r2. Pour des informations sur les systèmes ASA r2, voir "En savoir plus sur les systèmes de stockage ASA r2".

Créer un PVC en miroir

Suivez ces étapes et utilisez les exemples CRD pour créer une relation miroir entre les volumes primaires et secondaires.

Étapes

1. Effectuez les étapes suivantes sur le cluster Kubernetes principal :
   a. Créez un objet StorageClass 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:
        backendType: "ontap-nas"
        fsType: "nfs"
        trident.netapp.io/replication: "true"
      ```

   b. Créez un PVC avec le StorageClass précédemment créé.

      Exemple

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

   c. Créez un CR 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 actuel de protection des données (DP) du volume, puis remplit le champ d'état du MirrorRelationship.

d. Obtenez le TridentMirrorRelationship CR pour obtenir le nom interne et le 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 un StorageClass 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 un CR 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 créera une relation SnapMirror avec le nom de stratégie de relation configuré (ou par défaut pour ONTAP) et l'initialisera.

c. Créez un PVC avec la StorageClass précédemment créée pour servir de SnapMirror destination secondaire.

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 vérifiera la présence de la TridentMirrorRelationship CRD et ne créera pas le volume si la relation n'existe pas. Si la relation existe, Trident s'assurera que le nouveau volume FlexVol est placé sur une SVM appairée avec la SVM distante définie dans la MirrorRelationship.

États de réplication du volume

Une Trident Mirror Relationship (TMR) est une CRD qui représente une extrémité d'une relation de réplication entre des PVC. La TMR de destination possède un état, qui indique à Trident quel est l'état souhaité. La TMR de destination possède 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 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 a déjà été 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 en relation avec la source.

Promouvoir le PVC secondaire lors d'un basculement imprévu

Effectuez l'étape suivante sur le cluster Kubernetes secondaire :

- Mettez à jour le champ *spec.state* de TridentMirrorRelationship à `promoted`.

Promouvoir le PVC secondaire lors d'un basculement planifié

Lors d'un basculement (migration) planifié, effectuez les étapes suivantes pour promouvoir le PVC secondaire :

Étapes

1. Sur le cluster Kubernetes principal, créez un instantané du PVC et attendez que l'instantané soit créé.
2. Sur le cluster Kubernetes principal, créez le CR SnapshotInfo pour obtenir les détails 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* de la ressource personnalisée *TridentMirrorRelationship* à *promoted* et *spec.promotedSnapshotHandle* pour qu'il soit le internalName de l'instantané.
4. Sur le cluster Kubernetes secondaire, confirmez l'état (champ status.state) de TridentMirrorRelationship sur promu.

Rétablir une relation miroir après un basculement

Avant de rétablir une relation miroir, choisissez le côté que vous souhaitez définir comme nouveau principal.

Étapes

1. Sur le cluster Kubernetes secondaire, assurez-vous que les valeurs du champ *spec.remoteVolumeHandle* sur le TridentMirrorRelationship sont mises à jour.
2. Sur le cluster Kubernetes secondaire, mettez à jour le champ *spec.mirror* de TridentMirrorRelationship à `reestablished`.

Opérations supplémentaires

Trident prend en charge les opérations suivantes sur les volumes primaires et secondaires :

Répliquer le PVC primaire vers un nouveau PVC secondaire

Assurez-vous de disposer déjà d'un PVC primaire et d'un PVC secondaire.

Étapes

1. Supprimez les CRD PersistentVolumeClaim et TridentMirrorRelationship du cluster secondaire (de destination) établi.
2. Supprimez le CRD TridentMirrorRelationship du cluster principal (source).
3. Créez un nouveau TridentMirrorRelationship CRD sur le cluster principal (source) pour le nouveau PVC secondaire (destination) que vous souhaitez établir.

Redimensionner un PVC miroir, principal ou secondaire

Le PVC peut être redimensionné normalement, ONTAP étendra automatiquement les flevxols de destination si la quantité de données dépasse la taille actuelle.

Supprimer la réplication d'un PVC

Pour supprimer la réplication, effectuez l'une des opérations suivantes sur le volume secondaire actuel :

- Supprimez le MirrorRelationship sur le PVC secondaire. Cela interrompt la relation de réplication.
- Ou, mettez à jour le champ spec.state à *promoted*.

Supprimer un PVC (qui était précédemment mis en miroir)

Trident vérifie la présence de PVC répliqués et libère la relation de réplication avant de tenter de supprimer le volume.

Supprimer un TMR

La suppression d'un TMR sur l'un des côtés d'une relation miroir entraîne la transition du TMR restant à l'état *promoted* avant que Trident ne finalise la suppression. Si le TMR sélectionné pour suppression est déjà à l'état *promoted*, il n'existe aucune relation miroir et le TMR sera supprimé et Trident promouvra le PVC local à *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 devra utiliser un nouveau TMR avec un état de réplication de volume *established* lors de la création de la nouvelle relation miroir.

Mettez à jour les relations miroir lorsque ONTAP est en ligne

Les relations de miroir peuvent être mises à jour à tout moment après leur établissement. Vous pouvez utiliser le `state: promoted` ou `state: reestablished` champ pour mettre à jour les relations. Lors de la promotion d'un volume de destination vers un volume ReadWrite standard, vous pouvez utiliser *promotedSnapshotHandle* pour spécifier un instantané précis auquel restaurer le volume actuel.

Mettre à jour les relations miroir lorsque ONTAP est hors ligne

Vous pouvez utiliser une CRD pour effectuer une mise à jour SnapMirror sans que Trident ait de connexion directe au cluster ONTAP. Consultez l'exemple de format suivant de TridentActionMirrorUpdate :

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 de la TridentActionMirrorUpdate CRD. Elle peut prendre une valeur parmi *Succeeded*, *In Progress* ou *Failed*.

Utiliser la topologie CSI

Trident peut créer et attacher sélectivement des volumes aux nœuds présents dans un cluster Kubernetes en utilisant le "Fonctionnalité de topologie CSI".

Aperçu

Grâce à la fonctionnalité CSI Topology, l'accès aux volumes peut être limité à un sous-ensemble de nœuds, en fonction des régions et des zones de disponibilité. Les fournisseurs de cloud permettent aujourd'hui aux administrateurs Kubernetes de déployer des nœuds basés sur les zones. Les nœuds peuvent être situés dans différentes zones de disponibilité au sein d'une région, ou répartis sur diverses régions. Pour faciliter le provisionnement des volumes pour les charges de travail dans une architecture multizone, Trident utilise CSI Topology.

ASTUCE	Apprenez-en davantage sur la fonctionnalité de topologie CSI "ici".

Kubernetes propose deux modes de liaison de volumes uniques :

- Avec `VolumeBindingMode` défini sur `Immediate`, Trident crée le volume sans aucune prise en compte de la topologie. La liaison de volume et le provisionnement dynamique sont gérés lors de la création du PVC. C'est le comportement par défaut `VolumeBindingMode` et il convient aux clusters qui n'imposent pas de contraintes de topologie. Les volumes persistants sont créés sans dépendre des exigences de planification du pod demandeur.
- Avec `VolumeBindingMode` défini sur `WaitForFirstConsumer`, la création et la liaison d'un volume persistant pour un PVC sont différées jusqu'à ce qu'un pod utilisant ce PVC soit planifié et créé. De cette façon, les volumes sont créés pour répondre aux contraintes de planification imposées par les exigences

de topologie.

REMARQUE	Le `WaitForFirstConsumer` mode de liaison ne nécessite pas d'étiquettes de topologie. Cela peut être utilisé indépendamment de la fonctionnalité de topologie CSI.

Ce dont vous aurez besoin

Pour utiliser la topologie CSI, vous avez besoin 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 comporter des étiquettes permettant de prendre en compte la topologie (`topology.kubernetes.io/region` et `topology.kubernetes.io/zone`). Ces étiquettes **doivent être présentes sur les nœuds du cluster** avant l'installation de Trident pour que Trident puisse prendre en compte 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 : Créez un backend tenant compte de la topologie

Les backends de stockage Trident peuvent être conçus pour provisionner sélectivement des volumes en fonction des zones de disponibilité. Chaque backend peut comporter un bloc `supportedTopologies` optionnel qui représente une liste de zones et de régions prises en charge. Pour les StorageClasses qui utilisent un tel backend, un volume ne serait créé que si la demande provient d'une application planifiée dans une région ou une zone prise en charge.

Voici un exemple de définition de backend :

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 backend. Ces régions et zones représentent la liste des valeurs autorisées qui peuvent être fournies dans un StorageClass. Pour les StorageClasses qui contiennent un sous-ensemble des régions et zones fournies dans un backend, Trident crée un volume sur le backend.

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, les `region` et `zone` étiquettes représentent l'emplacement du pool de stockage. `topology.kubernetes.io/region` et `topology.kubernetes.io/zone` indiquent où les pools de stockage peuvent être utilisés.

Étape 2 : Définir les StorageClasses qui sont conscients de la topologie

En fonction des étiquettes de topologie fournies aux nœuds du cluster, il est possible de définir des StorageClasses contenant des informations de topologie. Cela déterminera les pools de stockage qui servent de candidats pour les requêtes PVC effectuées, ainsi que le sous-ensemble de nœuds pouvant 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 StorageClass fournie ci-dessus, `volumeBindingMode` est définie sur `WaitForFirstConsumer`. Les PVC demandés avec cette StorageClass ne seront pas traités tant qu'ils ne seront pas référencés dans un pod. Et, `allowedTopologies` fournit les zones et la région à utiliser. La `netapp-san-us-east1` StorageClass crée des PVC sur le backend `san-backend-us-east1` défini ci-dessus.

Étape 3 : Créer et utiliser un PVC

Une fois le StorageClass créé et associé à un backend, vous pouvez désormais créer des PVC.

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'un PVC à l'aide de ce manifeste donnerait le résultat suivant :

```
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 au PVC, utilisez le PVC 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 indique à Kubernetes de planifier le pod sur les nœuds présents dans la `us-east1` région, et de choisir parmi tous les nœuds présents dans la `us-east1-a` ou `us-east1-b` zones.

Voir la sortie suivante :

```
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
```

Mettre à jour les backends pour inclure `supportedTopologies`

Les backends préexistants peuvent être mis à jour pour inclure une liste de `supportedTopologies` en utilisant `tridentctl backend update`. Cela n'affectera pas les volumes déjà provisionnés et ne sera utilisé que pour les PVC ultérieurs.

Pour plus d'informations

- "Gérer les ressources pour les conteneurs"
- "nodeSelector"
- "Affinité et anti-affinité"
- "Souillures et tolérances"

Travailler avec des snapshots

Les snapshots de volumes persistants (PV) Kubernetes permettent de créer des copies à un instant précis des volumes. Vous pouvez créer un snapshot d'un volume créé avec Trident, importer un snapshot créé en dehors de Trident, créer un nouveau volume à partir d'un snapshot existant, et récupérer des données de volume à partir de snapshots.

Aperçu

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` pilotes.

Avant de commencer

Vous devez disposer d'un contrôleur de snapshots externe et de Custom Resource Definitions (CRDs) pour travailler avec les snapshots. C'est la responsabilité de l'orchestrateur Kubernetes (par exemple : Kubeadm, GKE, OpenShift).

Si votre distribution Kubernetes n'inclut pas le contrôleur de snapshots et les CRD, reportez-vous à Déployer un contrôleur d'instantané de volume.

REMARQUE	Ne créez pas de contrôleur de snapshots si vous créez des snapshots de volumes à la demande dans un environnement GKE. GKE utilise un contrôleur de snapshots intégré et masqué.

Créer un instantané de volume

Étapes

1. Créez un `VolumeSnapshotClass`. Pour plus d'informations, consultez "VolumeSnapshotClass".
   - Le `driver` pointe vers le pilote Trident CSI.
   - `deletionPolicy` peut être `Delete` ou `Retain`. Lorsqu'il est défini sur `Retain`, l'instantané physique sous-jacent sur le cluster de stockage est conservé même lorsque l'objet `VolumeSnapshot` est 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 instantané d'un PVC existant.

   Exemples

   - Cet exemple crée un instantané d'un PVC 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 d'instantané de volume pour un PVC nommé `pvc1` et le nom de l'instantané est défini sur `pvc1-snap`. Un VolumeSnapshot est analogue à un PVC et est associé à un `VolumeSnapshotContent` objet qui représente l'instantané 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 l' `VolumeSnapshotContent`objet pour le `pvc1-snap` VolumeSnapshot en le décrivant. Le `Snapshot Content Name` identifie l'objet VolumeSnapshotContent qui sert cet instantané. Le `Ready To Use` paramètre indique que l'instantané peut être utilisé pour créer un nouveau PVC.

```
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 un PVC à partir d'un instantané de volume

Vous pouvez utiliser `dataSource` pour créer un PVC en utilisant un VolumeSnapshot nommé `<pvc-name>` comme source des données. Après la création du PVC, il peut être attaché à un pod et utilisé comme n'importe quel autre PVC.

AVERTISSEMENT	Le PVC sera créé dans le même backend que le volume source. Consultez "KB : La création d'un PVC à partir d'un instantané de PVC Trident ne peut pas être effectuée dans un autre backend".

L'exemple suivant crée le PVC en utilisant `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 de snapshot pré-provisionné Kubernetes" pour permettre à l'administrateur du cluster de créer un `VolumeSnapshotContent` objet et d'importer des instantanés créés en dehors de Trident.

Avant de commencer

Trident doit avoir créé ou importé le volume parent de la capture.

Étapes

1. **Administrateur du cluster :** Créez un `VolumeSnapshotContent` objet qui référence l'instantané du backend. Cela lance le workflow d'instantané dans Trident.
   - Spécifiez le nom de l'instantané du backend 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 générateur de snapshots externe dans l'appel `ListSnapshots`.

REMARQUE	Le `<volumeSnapshotContentName>` ne peut pas toujours correspondre au nom de l'instantané du backend en raison des contraintes de dénomination des CR.

     Exemple

     L'exemple suivant crée un `VolumeSnapshotContent` objet qui référence l'instantané du backend `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. **Administrateur du cluster :** Créez la `VolumeSnapshot` CR qui référence l' `VolumeSnapshotContent`objet. Cela demande l'autorisation d'utiliser l' `VolumeSnapshot`dans un espace de noms donné.

   Exemple

   L'exemple suivant crée un `VolumeSnapshot` CR nommé `import-snap` qui référence le `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 gestionnaire de snapshots externe reconnaît le nouvellement créé `VolumeSnapshotContent` et exécute l'appel `ListSnapshots`. Trident crée le `TridentSnapshot`.
   - Le dispositif de capture d'écran externe définit le `VolumeSnapshotContent` sur `readyToUse` et le `VolumeSnapshot` sur `true`.
   - Trident retourne `readyToUse=true`.
4. **Tout utilisateur :** Créez un `PersistentVolumeClaim` pour référencer le nouveau `VolumeSnapshot`, où le `spec.dataSource` (ou `spec.dataSourceRef`) nom est le `VolumeSnapshot` nom.

Exemple

L'exemple suivant crée un PVC faisant référence au `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
```

Récupérer les données du volume à l'aide de snapshots

Le répertoire des instantanés est masqué par défaut afin d'assurer une compatibilité maximale des volumes provisionnés à l'aide des `ontap-nas` et `ontap-nas-economy` pilotes. Activez le répertoire `.snapshot` pour récupérer directement les données à partir des instantanés.

Utilisez la commande ONTAP volume snapshot restore pour restaurer un volume à un état enregistré dans un instantané précédent.

```
cluster1::*> volume snapshot restore -vserver vs0 -volume vol3 -snapshot
vol3_snap_archive
```

REMARQUE	Lorsque vous restaurez une copie instantanée, la configuration du volume existant est écrasée. Les modifications apportées aux données du volume après la création de la copie instantanée sont perdues.

Restauration sur place du volume à partir d'un instantané

Trident permet une restauration rapide et directe des volumes à partir d'un instantané grâce à la `TridentActionSnapshotRestore` (TASR) CR. Cette CR fonctionne comme une action Kubernetes impérative et n'est pas conservée après la fin de l'opération.

Trident prend en charge la restauration d'instantanés sur les `ontap-san`, `ontap-san-economy`, `ontap-nas`, `ontap-nas-flexgroup`, `azure-netapp-files`, `google-cloud-netapp-volumes` et `solidfire-san` pilotes.

Avant de commencer

Vous devez disposer d'une PVC reliée et d'un instantané de volume disponible.

- Vérifiez que le statut du PVC est bound.

```
kubectl get pvc
```

- Vérifiez que l'instantané du volume est prêt à être utilisé.

```
kubectl get vs
```

Étapes

1. Créez la TASR CR. Cet exemple crée une CR pour PVC `pvc1` et volume snapshot `pvc1-snapshot`.

   REMARQUE Le TASR CR doit se trouver dans un espace de noms 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 restaurer à partir de l'instantané. Cet exemple restaure à partir de l'instantané `pvc1`.

```
kubectl create -f tasr-pvc1-snapshot.yaml
```

```
tridentactionsnapshotrestore.trident.netapp.io/trident-snap created
```

Résultats

Trident restaure les données à partir de l'instantané. Vous pouvez vérifier l'état de la restauration de l'instantané :

```
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 relancera pas automatiquement l'opération en cas d'échec. Vous devrez effectuer l'opération à nouveau.
- Les utilisateurs de Kubernetes ne disposant pas d'un accès administrateur peuvent devoir obtenir l'autorisation de l'administrateur pour créer un TASR CR dans l'espace de noms de leur application.

Supprimez un PV avec les instantanés associés

Lors de la suppression d'un volume persistant avec des snapshots associés, le volume Trident correspondant passe à l'état « Suppression en cours ». Supprimez les snapshots du volume pour supprimer le volume Trident.

Déployer un contrôleur d'instantané de volume

Si votre distribution Kubernetes n'inclut pas le snapshot controller et les CRDs, vous pouvez les déployer comme suit.

Étapes

1. Créer des CRD d'instantané 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. Créez 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, ouvrez `deploy/kubernetes/snapshot-controller/rbac-snapshot-controller.yaml` et mettez à jour `namespace` pour votre espace de noms.

Liens associés

- "Instantanés de volume"
- "VolumeSnapshotClass"

Travailler avec les instantanés de groupe de volumes

Les instantanés de groupes de volumes Kubernetes des volumes persistants (PV) NetApp Trident offrent la possibilité de créer des instantanés de plusieurs volumes (un groupe d'instantanés de volumes). Cet instantané de groupe de volumes représente des copies de plusieurs volumes prises au même moment.

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 avec les pilotes de stockage suivants :

- `ontap-san` driver - 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 NetApp ASA r2 ou les systèmes de stockage AFX.

Avant de commencer

- Assurez-vous que votre version de Kubernetes est K8s 1.32 ou supérieure.
- Vous devez disposer d'un contrôleur de snapshots externe et de Custom Resource Definitions (CRDs) pour travailler avec les snapshots. C'est la responsabilité de l'orchestrateur Kubernetes (par exemple : Kubeadm, GKE, OpenShift).

  Si votre distribution Kubernetes n'inclut pas le contrôleur de snapshots externe et les CRD, reportez-vous à Déployer un contrôleur d'instantané de volume.

REMARQUE	Ne créez pas de contrôleur de snapshots si vous créez des snapshots de groupes de volumes à la demande dans un environnement GKE. GKE utilise un contrôleur de snapshots intégré et masqué.

- Dans le fichier YAML du contrôleur d'instantané, définissez la `CSIVolumeGroupSnapshot`feature gate sur « true » pour garantir que l'instantané du groupe de volumes est activé.
- Créez les classes d'instantané 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, consultez "VolumeGroupSnapshotClass".

```
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 en utilisant les classes de stockage existantes, ou ajoutez ces étiquettes aux PVC existants.

L'exemple suivant crée le PVC en utilisant `pvc1-group-snap` comme source de données et l'étiquette `consistentGroupSnapshot: groupA`. 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
```

- Créez un VolumeGroupSnapshot avec la même étiquette (`consistentGroupSnapshot: groupA`) spécifiée 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 les données du volume à l'aide d'un instantané de groupe

Vous pouvez restaurer des volumes persistants individuels à l'aide des instantanés individuels qui ont été créés dans le cadre de l'instantané de groupe de volumes. Vous ne pouvez pas restaurer l'instantané de groupe de volumes en tant qu'unité.

Utilisez la commande ONTAP volume snapshot restore pour restaurer un volume à un état enregistré dans un instantané précédent.

```
cluster1::*> volume snapshot restore -vserver vs0 -volume vol3 -snapshot
vol3_snap_archive
```

REMARQUE	Lorsque vous restaurez une copie instantanée, la configuration du volume existant est écrasée. Les modifications apportées aux données du volume après la création de la copie instantanée sont perdues.

Restauration sur place du volume à partir d'un instantané

Trident permet une restauration rapide et directe des volumes à partir d'un instantané grâce à la `TridentActionSnapshotRestore` (TASR) CR. Cette CR fonctionne comme une action Kubernetes impérative et n'est pas conservée après la fin de l'opération.

Pour plus d'informations, consultez "Restauration sur place du volume à partir d'un instantané".

Supprimer un PV avec des snapshots de groupe associés

Lors de la suppression d'un instantané de volume de groupe :

- Vous pouvez supprimer VolumeGroupSnapshots dans leur ensemble, pas les instantanés individuels du groupe.
- Si des PersistentVolumes sont supprimés alors qu'un instantané existe pour ce PersistentVolume, Trident déplacera ce volume vers un état « en cours 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 le clone commencera et le groupe ne pourra pas être supprimé tant que la division n'est pas terminée.

Déployer un contrôleur d'instantané de volume

Si votre distribution Kubernetes n'inclut pas le snapshot controller et les CRDs, vous pouvez les déployer comme suit.

Étapes

1. Créer des CRD d'instantané 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. Créez 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, ouvrez `deploy/kubernetes/snapshot-controller/rbac-snapshot-controller.yaml` et mettez à jour `namespace` pour votre espace de noms.

Liens associés

- "VolumeGroupSnapshotClass"
- "Instantanés de volume"

Gérer et surveiller Trident

Mettre à niveau Trident

Mettre à niveau Trident

À compter de la version 24.02, Trident adopte un cycle de publication de quatre mois, avec trois mises à jour majeures chaque année civile. Chaque nouvelle version s'appuie sur les précédentes et propose de nouvelles fonctionnalités, des améliorations de performances, des corrections de bogues et des améliorations. Nous vous encourageons à effectuer une mise à jour au moins une fois par an afin de profiter des nouvelles fonctionnalités de Trident.

Considérations avant la mise à niveau

Lors de la mise à niveau vers la dernière version de Trident, tenez compte des points suivants :

- Une seule instance de Trident doit être installée dans tous les espaces de noms d'un cluster Kubernetes donné.
- Trident 23.07 et versions ultérieures nécessitent des instantanés de volume v1 et ne prennent plus en charge les instantanés alpha ou beta.
- Lors de la mise à niveau, il est important de fournir `parameter.fsType` dans `StorageClasses` utilisé par Trident. Vous pouvez supprimer et recréer `StorageClasses` sans interrompre les volumes existants.
  - Il s'agit d'une **exigence** pour appliquer "contextes de sécurité" aux volumes SAN.
  - Le répertoire sample input contient des exemples, tels que `storage-class-basic.yaml.templ` et `storage-class-bronze-default.yaml`.
  - Pour plus d'informations, consultez "Problèmes connus".

Étape 1 : Sélectionnez une version

Les versions de Trident suivent une `YY.MM` convention de nommage basée sur la date, où « YY » sont les deux derniers chiffres de l'année et « MM » 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 24.06 (ou de toute version dérivée de 24.06) vers 25.06.
- Si vous effectuez une mise à niveau depuis une version en dehors de la fenêtre de quatre versions, procédez à 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 23.07 et souhaitez passer à la version 25.06 :
  a. Première mise à niveau de 23.07 à 24.06.
  b. Ensuite, passez de 24.06 à 25.06.

REMARQUE	Lors de la mise à niveau à l'aide de l'opérateur Trident sur OpenShift Container Platform, vous devez passer à Trident 21.01.1 ou une version ultérieure. L'opérateur Trident publié avec la version 21.01.0 contient un problème connu qui a été corrigé dans la version 21.01.1. Pour plus de détails, consultez le "détails du problème sur GitHub".

Étape 2 : Déterminez la méthode d'installation d'origine

Pour déterminer la version que vous avez utilisée pour installer Trident :

1. Utilisez `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 pod opérateur, Trident a été installé à l'aide de l'opérateur Trident, soit manuellement, soit à l'aide de Helm.
2. S'il existe un pod opérateur, utilisez `kubectl describe torc` pour déterminer si Trident a été installé à l'aide de Helm.
   - S'il y a une étiquette Helm, Trident a été installé avec 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 employée lors de l'installation initiale, cependant vous pouvez "passer d'une méthode d'installation à une 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 vérifier "Comprendre le workflow de mise à niveau de l'opérateur" avant de procéder à 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 lors de la mise à niveau. Cela inclut les modifications apportées au contrôleur Trident, au pod du contrôleur, aux pods du nœud et au DaemonSet du nœud qui permettent les mises à jour progressives.

Gestion de la mise à niveau de 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 progressives*". En particulier, l'opérateur Trident communique avec le cluster Kubernetes pour :

- Supprimez et recréez le déploiement du contrôleur Trident et le DaemonSet du nœud.

- Remplacez le Trident Controller Pod et les Trident Node Pods par de nouvelles versions.
  - Si un nœud n'est pas mis à jour, cela n'empêche pas les autres nœuds d'être mis à jour.
  - Seuls les nœuds avec un Trident Node Pod en cours d'exécution peuvent monter des volumes.

ASTUCE	Pour plus d'informations sur l'architecture Trident sur le cluster Kubernetes, veuillez vous référer à "Architecture Trident".

Flux de travail de mise à niveau de l'opérateur

Lorsque vous lancez une mise à niveau à l'aide de l'opérateur Trident :

1. L'opérateur **Trident** :
   a. Détecte la version actuellement installée de Trident (version *n*).
   b. Met à jour tous les objets Kubernetes, y compris les CRDs, RBAC et Trident SVC.
   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 Trident Controller pour *n+1*.
3. L'opérateur **Trident** :
   a. Supprime le DaemonSet Trident Node pour *n*. L'opérateur n'attend pas la fin du Pod Node.
   b. Crée le Daemonset de nœud Trident pour *n+1*.
4. **Kubernetes** crée des pods de nœud Trident sur les nœuds qui n'exécutent pas le pod de nœud Trident *n*. Cela garantit qu'il n'y a jamais plus d'un pod de nœud Trident, de n'importe quelle 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 l'échelle d'un cluster.

REMARQUE	Pour mettre à niveau Trident installé à l'aide de l'opérateur à portée d'espace de noms (versions 20.07 à 20.10), utilisez les instructions de mise à niveau "votre version installée" de Trident.

À propos de cette tâche

Trident fournit un fichier bundle que vous pouvez utiliser pour installer l'opérateur et créer les objets associés pour votre version de Kubernetes.

- Pour les clusters exécutant Kubernetes 1.25 ou une version ultérieure, utilisez "bundle_post_1_25.yaml".

Avant de commencer

Assurez-vous d'utiliser un cluster Kubernetes exécutant "une version Kubernetes prise en charge".

Étapes

1. Vérifiez votre version de Trident :

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

2. Mettez 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), ainsi que 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 depuis 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 des `TridentOrchestrator` attributs, vous pouvez modifier l'objet `TridentOrchestrator` pour modifier les paramètres d'installation. Cela peut inclure des modifications apportées pour spécifier des registres d'images Trident et CSI en miroir pour le mode hors ligne, activer les journaux de débogage ou spécifier des secrets d'extraction d'images.
5. Installez Trident en utilisant le fichier YAML de bundle approprié pour votre environnement, où *<bundle.yaml>* est `bundle_pre_1_25.yaml` ou `bundle_post_1_25.yaml` selon 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 trident torc 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 à 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 `--set excludePodSecurityPolicy=true` à la commande `helm upgrade` avant de pouvoir mettre à niveau le cluster.

Si vous avez déjà mis à niveau votre cluster Kubernetes de la version 1.24 à la version 1.25 sans mettre à niveau le helm Trident, la mise à niveau du helm échouera. Pour que la mise à niveau du helm réussisse, effectuez les étapes suivantes comme prérequis :

1. Installez le plugin helm-mapkubeapis depuis https://github.com/helm/helm-mapkubeapis.

2. Effectuez un test à blanc pour la version Trident dans l'espace de noms où Trident est installé. Cela liste les ressources qui seront nettoyées.

```
helm mapkubeapis --dry-run trident  --namespace trident
```

3. Effectuez une exécution complète avec helm pour effectuer le nettoyage.

```
helm mapkubeapis trident  --namespace trident
```

Étapes

1. Si vous "installé Trident à l'aide de Helm" , 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 depuis "la section *Assets* sur GitHub".
   b. Utilisez la commande `helm upgrade` où `trident-operator-26.02.0.tgz` correspond à la version vers laquelle vous souhaitez effectuer la mise à niveau.

```
helm upgrade <name> trident-operator-26.02.0.tgz
```

REMARQUE	Si vous définissez des options personnalisées lors de l'installation initiale (comme la spécification de registres privés et en miroir pour Trident et les images CSI), ajoutez la `helm upgrade` commande en utilisant `--set` pour garantir que ces options sont incluses dans la commande de mise à niveau, sinon les valeurs seront réinitialisées à la valeur par défaut.

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

Mise à niveau d'une `tridentctl`installation vers l'opérateur Trident

Vous pouvez mettre à niveau vers la dernière version de l'opérateur Trident à partir d'une installation `tridentctl` existante. Les backends et les PVC existants seront automatiquement disponibles.

REMARQUE	Avant de changer de méthode d'installation, consultez "Passer d'une méthode d'installation à une autre".

Étapes

1. Téléchargez la dernière version de Trident.

```
# Download the release required [26.02.0]
mkdir 26.02.0
cd 26.02.0
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
```

2. Créez la `tridentorchestrator` CRD à partir du manifeste.

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

3. Déployez l'opérateur à portée de cluster dans le même espace de noms.

```
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. Créer 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. Confirmez 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:                v26.02.0
```

Mise à niveau avec tridentctl

Vous pouvez facilement mettre à niveau une installation Trident existante en utilisant `tridentctl`.

À propos de cette tâche

La désinstallation et la réinstallation de Trident constituent une mise à niveau. Lorsque vous désinstallez Trident, la Persistent Volume Claim (PVC) et le Persistent Volume (PV) utilisés par le déploiement Trident ne sont pas supprimés. Les PV déjà provisionnés restent disponibles pendant que Trident est hors ligne, et Trident provisionnera des volumes pour toutes les PVC créées entre-temps après son retour en ligne.

Avant de commencer

Vérifiez "Sélectionnez une méthode de mise à niveau" avant la mise à niveau en utilisant `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. Consultez "Installez Trident à l'aide de tridentctl".

IMPORTANT	N'interrompez pas le processus de mise à niveau. Assurez-vous que l'installateur s'exécute jusqu'à la fin.

Gérez Trident à l'aide de tridentctl

Le "Trident bundle d'installation" inclut l' `tridentctl`utilitaire en ligne de commande pour fournir un accès simple à Trident. Les utilisateurs Kubernetes disposant de privilèges suffisants peuvent l'utiliser pour installer Trident ou gérer l'espace de noms qui contient le pod Trident.

Commandes et drapeaux globaux

Vous pouvez exécuter `tridentctl help` pour obtenir une liste des commandes disponibles pour `tridentctl` ou ajouter le drapeau `--help` à n'importe quelle commande pour obtenir une liste des options et des drapeaux 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 conteneurs nécessaires à Trident.

`import`
Importer une ressource existante dans Trident.

`install`
Installez Trident.

`logs`
Imprimez les journaux de Trident.

`send`
Envoyez une ressource depuis Trident.

`uninstall`
Désinstallez Trident.

`update`
Modifier une ressource dans Trident.

`update backend state`
Suspendre temporairement les opérations en arrière-plan.

`upgrade`
Mettre à niveau une ressource dans Trident.

`version`
Imprimez la version de Trident.

Drapeaux globaux

`-d, --debug`
Sortie de débogage.

`-h, --help`
Aide pour `tridentctl`.

`-k, --kubeconfig string`
Spécifiez le `KUBECONFIG` chemin pour exécuter des commandes localement ou d'un cluster Kubernetes à un autre.

REMARQUE	Vous pouvez également exporter la `KUBECONFIG` variable pour qu'elle pointe vers un cluster Kubernetes spécifique et émettre `tridentctl` commandes à ce cluster.

`-n, --namespace string`
Espace de noms du déploiement Trident.

`-o, --output string`
Format de sortie. L'un de json|yaml|name|wide|ps (par défaut).

`-s, --server string`
Adresse/port de l'interface REST de Trident.

AVERTISSEMENT	L'interface REST de Trident peut être configurée pour écouter et servir uniquement à 127.0.0.1 (pour IPv4) ou [::1] (pour IPv6).

Options de commande et drapeaux

créer

Utilisez la `create` commande pour ajouter une ressource à Trident.

```
tridentctl create [option]
```

Options

`backend`: Ajouter un backend à Trident.

supprimer

Utilisez la commande `delete` pour supprimer une ou plusieurs ressources de Trident.

```
tridentctl delete [option]
```

Options

`backend`: Supprimer un ou plusieurs backends de stockage de Trident.
`snapshot`: Supprimer un ou plusieurs snapshots de volume de Trident.

`storageclass`: Supprimer une ou plusieurs classes de stockage de Trident.
`volume`: Supprimer un ou plusieurs volumes de stockage de Trident.

obtenir

Utilisez la `get` commande pour obtenir une ou plusieurs ressources de Trident.

```
tridentctl get [option]
```

Options

`backend`: Obtenir un ou plusieurs backends de stockage depuis Trident.
`snapshot`: Obtenir un ou plusieurs snapshots depuis Trident.
`storageclass`: Obtenir une ou plusieurs classes de stockage depuis Trident.
`volume`: Obtenir un ou plusieurs volumes depuis Trident.

Drapeaux

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

images

Utilisez `images` les indicateurs pour afficher un tableau des images de conteneur dont Trident a besoin.

```
tridentctl images [flags]
```

Drapeaux

`-h`, `--help` : Aide pour les images.
`-v`, `--k8s-version string` : Version sémantique du cluster Kubernetes.

importer un volume

Utilisez la `import volume` commande pour importer un volume existant dans Trident.

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

Alias

`volume`, `v`

Drapeaux

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

installer

Utilisez les `install` flags pour installer Trident.

```
tridentctl install [flags]
```

Drapeaux

`--autosupport-image string` : L'image de conteneur pour Autosupport Telemetry (par défaut « netapp/trident autosupport:<current-version> »).
`--autosupport-proxy string` : L'adresse/port d'un proxy pour l'envoi de la télémétrie Autosupport.
`--enable-node-prep` : Tenter d'installer les paquets requis sur les nœuds.
`--generate-custom-yaml` : Générer des fichiers YAML sans rien installer.
`-h, --help` : Aide pour l'installation.
`--http-request-timeout` : Remplacer le délai d'expiration des requêtes HTTP pour l'API REST du contrôleur Trident (par défaut 1m30s).
`--image-registry string` : L'adresse/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 hôte de l'état interne de kubelet (par défaut « /var/lib/kubelet »).
`--log-format string` : Le format de journalisation de Trident (text, json) (par défaut « text »).
`--node-prep` : Permet à Trident de préparer les nœuds du cluster Kubernetes à gérer les volumes en utilisant le protocole de stockage des données spécifié. **Actuellement, `iscsi` est la seule valeur prise en charge. À partir de 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, assurez-vous qu'il n'existe pas (par défaut « trident »).
`--pvc string` : Le nom du PVC hérité utilisé par Trident, assurez-vous qu'il n'existe pas (par défaut « trident »).
`--silence-autosupport` : Ne pas envoyer automatiquement les bundles autosupport à NetApp (par défaut true).
`--silent` : Désactiver la plupart des sorties pendant l'installation.
`--trident-image string` : L'image Trident à installer.
`--k8s-api-qps` : La limite de requêtes par seconde (QPS) pour les requêtes API Kubernetes (par défaut 100 ; optionnel).
`--use-custom-yaml` : Utiliser tout fichier YAML existant dans le répertoire de configuration.
`--use-ipv6` : Utiliser IPv6 pour la communication de Trident.

journaux

Utilisez `logs` les indicateurs pour imprimer les journaux de Trident.

```
tridentctl logs [flags]
```

Drapeaux

`-a, --archive` : Créer une archive de support contenant tous les journaux, sauf indication contraire.
`-h, --help` : Aide pour les journaux.
`-l, --log string` : Journal Trident à afficher. Un de trident|auto|trident-operator|all (par défaut « auto »).
`--node string` : Le nom du nœud Kubernetes à partir duquel collecter les journaux des pods du nœud.
`-p, --previous` : Récupérer les journaux de l'instance précédente du conteneur si elle existe.
`--sidecars` : Récupérer les journaux des conteneurs sidecar.

envoyer

Utilisez la `send` commande pour envoyer une ressource depuis Trident.

```
tridentctl send [option]
```

Options

`autosupport`: Envoyer une archive Autosupport à NetApp.

désinstaller

Utilisez `uninstall` les options pour désinstaller Trident.

```
tridentctl uninstall [flags]
```

Drapeaux

`-h, --help`: Aide à la désinstallation.
`--silent`: Désactive la plupart des messages pendant la désinstallation.

mise à jour

Utilisez la `update` commande pour modifier une ressource dans Trident.

```
tridentctl update [option]
```

Options

`backend`: Mettre à jour un backend dans Trident.

mettre à jour l'état du backend

Utilisez la commande `update backend state` pour suspendre ou reprendre les opérations en arrière-plan.

```
tridentctl update backend state <backend-name> [flag]
```

Points à considérer

- Si un backend est créé à l'aide d'un 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 commande `tridentctl update backend state <backend-name> --user-state suspended/normal`.
- Pour retrouver la possibilité de définir le `userState` via tridentctl après qu'il a été défini via tbc, le champ `userState` doit être supprimé du tbc. Cela peut être fait en utilisant la commande `kubectl edit tbc`. Après la suppression du champ `userState`, vous pouvez utiliser la commande `tridentctl update backend state` pour modifier le `userState` d'un backend.
- Utilisez le `tridentctl update backend state` pour modifier le `userState`. Vous pouvez également mettre à jour le `userState` en utilisant `TridentBackendConfig` ou `backend.json` fichier ; cela déclenche une réinitialisation complète du backend et peut prendre du temps.

  #### Drapeaux

  `-h, --help`: Aide pour l'état du backend.
  `--user-state`: Définir sur `suspended` pour mettre en pause les opérations du backend. Définir sur `normal` pour reprendre les opérations du backend. Lorsque défini sur `suspended` :

- `AddVolume` and `Import Volume` sont en pause.
- `CloneVolume`, `ResizeVolume`, `PublishVolume`, `UnPublishVolume`, `CreateSnapshot`, `GetSnapshot`, `RestoreSnapshot`, `DeleteSnapshot`, `RemoveVolume`, `GetVolumeExternal`,

`ReconcileNodeAccess` restent disponibles.

Vous pouvez également mettre à jour l'état du backend à l'aide du champ `userState` dans le fichier de configuration du backend `TridentBackendConfig` ou `backend.json`. Pour plus d'informations, consultez "Options pour la gestion des backends" et "Effectuez la gestion du backend avec kubectl".

Exemple :

JSON

Suivez ces étapes pour mettre à jour le `userState` en utilisant le fichier `backend.json` :

1. Modifiez le `backend.json` fichier pour inclure le `userState` champ avec sa valeur définie sur 'suspended'.
2. Mettez à jour le backend en utilisant la `tridentctl update backend` commande et le chemin vers le fichier `backend.json` mis à jour.

   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 le tbc après son application à l'aide de la `kubectl edit <tbc-name> -n <namespace>` commande. L'exemple suivant met à jour l'état du backend pour le suspendre à l'aide de l'option `userState: suspended` :

```
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

Utilisez `version` les options pour afficher la version de `tridentctl` et le service Trident en cours d'exécution.

```
tridentctl version [flags]
```

Drapeaux

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

Prise en charge des plugins

Tridentctl prend en charge les plugins 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 listés dans la section plugin de l'aide de tridentctl. Vous pouvez également limiter la recherche en spécifiant un dossier de plugin dans la variable d'environnement TRIDENTCTL_PLUGIN_PATH (Example: `TRIDENTCTL_PLUGIN_PATH=~/tridentctl-plugins/`). Si la variable est utilisée, tridentctl recherche uniquement dans le dossier spécifié.

Surveiller Trident

Trident fournit un ensemble de points de terminaison de métriques Prometheus que vous pouvez utiliser pour surveiller les performances de Trident.

Aperçu

Les indicateurs fournis par Trident vous permettent de faire ce qui suit :

- Gardez un œil sur la santé et la configuration de Trident. Vous pouvez examiner le succès des opérations et vérifier si Trident peut communiquer avec les backends comme prévu.
- Examinez les informations d'utilisation du backend et comprenez combien de volumes sont provisionnés sur un backend et la quantité d'espace consommée, et ainsi de suite.
- Maintenir une cartographie de la quantité de volumes provisionnés sur les backends disponibles.
- Suivez les performances. Vous pouvez voir combien de temps il faut à Trident pour communiquer avec les backends et effectuer des opérations.

REMARQUE	Par défaut, les métriques de Trident sont exposées sur le port cible `8001` à l'endpoint `/metrics`. Ces métriques sont **activées par défaut** lors de l'installation de Trident. Vous pouvez également configurer la consommation des métriques Trident via HTTPS sur le port `8444`.

Ce dont vous aurez besoin

- Un cluster Kubernetes avec Trident installé.
- Une instance Prometheus. Il peut s'agir d'un "déploiement Prometheus conteneurisé" ou vous pouvez choisir d'exécuter 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 métriques Trident, vous devez créer un Prometheus ServiceMonitor qui surveille le `trident-csi` service et écoute sur le `metrics` port. Un exemple de ServiceMonitor ressemble à ceci :

```
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  name: trident-sm
  namespace: monitoring
  labels:
    release: prom-operator
spec:
  jobLabel: trident
  selector:
    matchLabels:
      app: controller.csi.trident.netapp.io
  namespaceSelector:
    matchNames:
      - trident
  endpoints:
    - port: metrics
      interval: 15s
```

Cette définition de ServiceMonitor récupère les métriques renvoyées par le service `trident-csi` et recherche spécifiquement le point de terminaison `metrics` du service. Par conséquent, Prometheus est désormais configuré pour comprendre les métriques de Trident.

Outre les métriques directement accessibles depuis Trident, kubelet expose de nombreuses `kubelet_volume_*` métriques via son propre point de terminaison de métriques. Kubelet peut fournir des informations sur les volumes attachés, les pods et d'autres opérations internes qu'il gère. Consultez "ici".

Consommer les métriques Trident via HTTPS

Pour consommer les métriques Trident via HTTPS (port 8444), vous devez modifier la définition ServiceMonitor afin d'inclure la configuration TLS. Vous devez également copier le `trident-csi` secret de l'`trident` espace de noms vers l'espace de noms où Prometheus est en cours d'exécution. Vous pouvez le faire à l'aide de la commande suivante :

```
kubectl get secret trident-csi -n trident -o yaml | sed 's/namespace:
trident/namespace: monitoring/' | kubectl apply -f -
```

Un exemple de ServiceMonitor pour les métriques HTTPS ressemble à ceci :

```
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, Helm chart et Operator :

- Si vous utilisez la commande `tridentctl install`, vous pouvez passer le paramètre `--https-metrics` pour activer les métriques HTTPS.
- Si vous utilisez le Helm chart, vous pouvez définir le paramètre `httpsMetrics` pour activer les métriques HTTPS.
- Si vous utilisez des fichiers YAML, vous pouvez ajouter le `--https_metrics` flag au `trident-main` container dans le `trident-deployment.yaml` fichier.

Étape 3 : Interroger les métriques Trident avec PromQL

PromQL est idéal pour créer des expressions qui renvoient des données de séries temporelles ou tabulaires.

Voici quelques requêtes PromQL que vous pouvez utiliser :

Obtenez des informations sur la santé de Trident

- **Pourcentage de réponses HTTP 2XX provenant 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 REST de Trident via 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 total de volume alloué par chaque backend**

```
sum (trident_volume_allocated_bytes) by (backend_uuid)
```

Obtenez l'utilisation individuelle du volume

REMARQUE	Ceci n'est activé que si les métriques kubelet sont également collectées.

- **Pourcentage d'espace utilisé pour chaque volume**

```
kubelet_volume_stats_used_bytes / kubelet_volume_stats_capacity_bytes *
100
```

Découvrez la télémétrie AutoSupport de Trident

Par défaut, Trident envoie quotidiennement des métriques Prometheus et des informations de base sur le backend à NetApp.

- Pour empêcher Trident d'envoyer des métriques Prometheus et des informations de base sur le backend à NetApp, transmettez le `--silence-autosupport` indicateur lors de l'installation de Trident.
- Trident peut également envoyer les journaux de conteneurs à l'assistance NetApp à la demande via `tridentctl send autosupport`. Vous devrez déclencher Trident pour qu'il télécharge ses journaux. Avant de soumettre les journaux, vous devez accepter les "politique de confidentialité" de NetApp.
- Sauf indication contraire, Trident récupère les journaux des dernières 24 heures.
- Vous pouvez spécifier la durée de conservation des journaux avec le `--since` indicateur. Par exemple : `tridentctl send autosupport --since=1h`. Ces informations sont collectées et envoyées via un `trident-autosupport` conteneur installé avec Trident. Vous pouvez obtenir l'image du conteneur à "Trident AutoSupport".
- 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 utile 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 point de terminaison AutoSupport de NetApp. Si vous utilisez un registre privé pour stocker les images de conteneur, vous pouvez utiliser l'indicateur `--image-registry`.
- Vous pouvez également configurer les URL de proxy en générant les fichiers YAML d'installation. Cela

peut être fait en utilisant `tridentctl install --generate-custom-yaml` pour créer les fichiers YAML et en ajoutant l'argument `--proxy-url` pour le conteneur `trident-autosupport` dans `trident-deployment.yaml`.

Désactiver les métriques Trident

Pour **désactiver** le signalement des métriques, vous devez générer des fichiers YAML personnalisés (à l'aide de l' `--generate-custom-yaml`flag) et les modifier pour supprimer le `--metrics`flag lors de l'appel pour le `trident-main`conteneur.

Désinstaller Trident

Vous devez utiliser la même méthode pour désinstaller Trident que celle que vous avez utilisée pour l'installer.

À propos de cette tâche

- Si vous avez besoin d'un correctif pour des bugs observés après une mise à niveau, des problèmes de dépendances ou une mise à niveau incomplète ou ayant échoué, vous devez désinstaller Trident et réinstaller la version précédente en utilisant les instructions spécifiques pour cela "version". Il s'agit de la seule méthode recommandée pour *downgrader* vers une version antérieure.
- Pour faciliter la mise à niveau et la réinstallation, la désinstallation de Trident ne supprime pas les CRD ni les objets associés créés par Trident. Si vous devez supprimer complètement Trident et toutes ses données, consultez "Supprimez complètement Trident et les CRD".

Avant de commencer

Si vous mettez hors service des clusters Kubernetes, vous devez supprimer toutes les applications qui utilisent des volumes créés par Trident avant de désinstaller. Cela garantit que les PVC sont dépubliés sur les nœuds Kubernetes avant leur suppression.

Déterminez la méthode d'installation d'origine

Vous devez utiliser la même méthode pour désinstaller Trident que celle que vous avez utilisée pour l'installer. Avant de désinstaller, vérifiez quelle version vous avez utilisée pour installer Trident à l'origine.

1. Utilisez `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 pod opérateur, Trident a été installé à l'aide de l'opérateur Trident, soit manuellement, soit à l'aide de Helm.
2. S'il existe un pod opérateur, utilisez `kubectl describe tproc trident` pour déterminer si Trident a été installé à l'aide de Helm.
   - S'il y a une étiquette Helm, Trident a été installé avec Helm.
   - S'il n'y a pas d'étiquette Helm, Trident a été installé manuellement à l'aide de l'opérateur Trident.

Désinstaller une installation de l'opérateur Trident

Vous pouvez désinstaller une installation de l'opérateur Trident manuellement ou à l'aide de Helm.

Désinstaller l'installation manuelle

Si vous avez installé Trident à l'aide de l'opérateur, vous pouvez le désinstaller en procédant de l'une des manières 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` flag est défini sur `true`, l'opérateur Trident désinstalle Trident, mais ne supprime pas le TridentOrchestrator lui-même. Vous devez nettoyer le TridentOrchestrator et en créer un nouveau si vous souhaitez installer à nouveau Trident.

2. **Supprimer `TridentOrchestrator`** : En supprimant le `TridentOrchestrator` CR qui a été utilisé pour déployer Trident, vous demandez à l'opérateur de désinstaller Trident. L'opérateur traite la suppression de `TridentOrchestrator` et procède à la suppression du déploiement Trident et du daemonset, supprimant les pods Trident qu'il avait créés lors de l'installation.

   ```
   kubectl delete -f deploy/<bundle.yaml> -n <namespace>
   ```

Désinstaller l'installation 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
```

Désinstaller une `tridentctl` installation

Utilisez la commande `uninstall` dans `tridentctl` pour supprimer toutes les ressources associées à Trident à l'exception des CRDs et des objets associés :

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

Trident for Docker

Prérequis pour le déploiement

Vous devez installer et configurer les prérequis de protocole nécessaires sur votre hôte avant de pouvoir déployer Trident.

Vérifiez les exigences

- Vérifiez que votre déploiement répond à toutes les "exigences".
- Vérifiez que vous disposez d'une version compatible de Docker. Si votre version de Docker est obsolète, "installer ou mettre à jour".

```
docker --version
```

- Vérifiez que les prérequis du protocole sont installés et configurés sur votre hôte.

Outils NFS

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

RHEL 8+

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

Ubuntu

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

AVERTISSEMENT	Redémarrez vos nœuds de travail après l'installation des outils NFS pour éviter les échecs lors de l'attachement des volumes aux conteneurs.

Outils iSCSI

Installez les outils iSCSI en utilisant les commandes pour votre système d'exploitation.

RHEL 8+

1. Installez les paquets système suivants :

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

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

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

3. Définissez la numérisation sur manuel :

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

4. Activez le multipathing :

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

REMARQUE	Assurez-vous `etc/multipath.conf` contient `find_multipaths no` sous `defaults`.

5. Assurez-vous 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 paquets système suivants :

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

2. Vérifiez que la version d'open-iscsi est 2.0.874-5ubuntu2.10 ou ultérieure (pour bionic) ou 2.0.874-7.1ubuntu6.1 ou ultérieure (pour focal):

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

3. Définissez la numérisation sur manuel :

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

4. Activez le multipathing :

```
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	Assurez-vous `etc/multipath.conf` contient `find_multipaths no` sous `defaults`.

5. Assurez-vous que `open-iscsi` et `multipath-tools` sont activés 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 pour votre système d'exploitation.

REMARQUE	• NVMe nécessite 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 vers une version 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 correspondant à votre système d'exploitation.

- Lors de l'utilisation de nœuds de travail exécutant RHEL/Red Hat Enterprise Linux CoreOS (RHCOS) avec des FC PVs, spécifiez le `discard` mountOption dans le StorageClass pour effectuer une récupération d'espace en ligne. Consultez "Documentation Red Hat".

RHEL 8+

1. Installez les paquets système suivants :

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

2. Activez le multipathing :

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

REMARQUE	Assurez-vous `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 paquets système suivants :

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

2. Activez le multipathing :

```
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	Assurez-vous `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
```

Déployer 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 vers les hôtes Docker, avec un cadre permettant l'ajout de plateformes supplémentaires à l'avenir.

Plusieurs instances de Trident peuvent s'exécuter simultanément sur le même hôte. Cela permet des connexions simultanées à plusieurs systèmes de stockage et types de stockage, avec la possibilité de personnaliser le stockage utilisé pour les volumes Docker.

Ce dont vous aurez besoin

Voir le "conditions préalables au déploiement". Une fois que vous vous êtes assuré que les prérequis sont remplis, vous êtes prêt à déployer Trident.

Méthode de plugin géré par Docker (version 1.13/17.03 et ultérieure)

REMARQUE	**Avant de commencer** Si vous avez utilisé Trident avant Docker 1.13/17.03 dans la méthode traditionnelle du démon, assurez-vous d'arrêter le processus Trident et de redémarrer votre démon Docker avant d'utiliser la méthode du plugin géré.

1. Arrêtez toutes les instances en cours d'exécution :

```
pkill /usr/local/bin/netappdvp
pkill /usr/local/bin/trident
```

2. Redémarrez Docker.

```
systemctl restart docker
```

3. Assurez-vous d'avoir installé Docker Engine 17.03 (new 1.13) ou une version ultérieure.

```
docker --version
```

Si votre version est obsolète "installer ou mettre 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 n'importe quel nom de votre choix en spécifiant l'option `config` avec le nom de fichier. Le fichier de configuration doit se trouver dans le répertoire `/etc/netappdvp` sur le système hôte.
   - `log-level` : Spécifiez le niveau de journalisation (`debug`, `info`, `warn`, `error`, `fatal`). La valeur par défaut est `info`.

- `debug`: Indiquez si la journalisation de débogage est activée. La valeur par défaut est false. Remplace log-level si true.

  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 en utilisant le système de plugins gérés. Remplacez `<version>` par la version du plugin (xxx.xx.x) que vous utilisez.

   ```
   docker plugin install --grant-all-permissions --alias netapp
   netapp/trident-plugin:<version> config=myConfigFile.json
   ```

3. Commencez à utiliser Trident pour consommer du stockage à partir du système configuré.

   a. Créez un volume nommé « firstVolume » :

   ```
   docker volume create -d netapp --name firstVolume
   ```

   b. Créer 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. Assurez-vous d'avoir Docker version 1.10 ou ultérieure.

   ```
   docker --version
   ```

   Si votre version est obsolète, mettez à jour votre installation.

   ```
   curl -fsSL https://get.docker.com/ | sh
   ```

   Ou, "Suivez les instructions pour votre distribution".

2. Assurez-vous que NFS et/ou iSCSI sont configurés pour votre système.

Étapes

1. Installez et configurez le plugin Docker Volume NetApp :
   a. Téléchargez et décompressez 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. Accédez à un emplacement dans le chemin du bin :

   ```
   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".

Après le démarrage du démon, vous pouvez créer et gérer des volumes à l'aide de l'interface CLI Docker.

3. Créer un volume :

```
docker volume create -d netapp --name trident_1
```

4. Provisionnez 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
```

Démarrer Trident au démarrage du système

Un exemple de fichier d'unité pour les systèmes basés sur systemd est disponible à `contrib/trident.service.example` dans le dépôt Git. 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 vous avez plus d'une instance en cours d'exécution.

   ```
   cp contrib/trident.service.example
   /usr/lib/systemd/system/trident.service
   ```

2. Modifiez le fichier, changez la description (ligne 2) pour qu'elle corresponde au nom du pilote et le chemin du fichier de configuration (ligne 9) pour qu'il reflète votre environnement.
3. Rechargez systemd pour qu'il prenne en compte les modifications :

   ```
   systemctl daemon-reload
   ```

4. Activez le service.

   Ce nom varie en fonction du nom que vous avez donné au fichier dans le `/usr/lib/systemd/system` répertoire.

   ```
   systemctl enable trident
   ```

5. Démarrez le service.

   ```
   systemctl start trident
   ```

6. Consultez le statut.

   ```
   systemctl status trident
   ```

REMARQUE	Chaque fois que vous modifiez le fichier d'unité, exécutez la commande `systemctl daemon-reload` pour qu'il prenne en compte les modifications.

Mettre à niveau ou désinstaller Trident

Vous pouvez mettre à jour Trident pour Docker en toute sécurité, sans impact sur les volumes en cours d'utilisation. Pendant le processus de mise à jour, il y aura une brève période où `docker volume` les commandes adressées au plugin ne réussiront pas, et les applications ne pourront pas monter les volumes tant que le plugin ne sera pas de nouveau opérationnel. Dans la plupart des cas, cela ne dure que quelques secondes.

Mise à niveau

Effectuez les étapes ci-dessous pour mettre à niveau Trident pour Docker.

Étapes

1. Répertoriez les volumes existants :

```
docker volume ls
DRIVER              VOLUME NAME
netapp:latest       my_volume
```

2. Désactivez le plugin :

```
docker plugin disable -f netapp:latest
docker plugin ls
ID                  NAME                DESCRIPTION
ENABLED
7067f39a5df5        netapp:latest       nDVP - NetApp Docker Volume
Plugin   false
```

3. Mettre à jour le plugin :

```
docker plugin upgrade --skip-remote-check --grant-all-permissions
netapp:latest netapp/trident-plugin:21.07
```

REMARQUE	La version 18.01 de Trident remplace le nDVP. Vous devez effectuer la mise à niveau directement de l' `netapp/ndvp-plugin` image vers l' `netapp/trident-plugin` image.

4. Activez le plugin :

```
docker plugin enable netapp:latest
```

5. Vérifiez que le plugin est activé :

```
docker plugin ls
ID                  NAME                DESCRIPTION
ENABLED
7067f39a5df5        netapp:latest       Trident - NetApp Docker Volume
Plugin   true
```

6. Vérifiez que les volumes sont visibles:

```
docker volume ls
DRIVER              VOLUME NAME
netapp:latest       my_volume
```

IMPORTANT

Si vous effectuez une mise à niveau depuis une ancienne version de Trident (pré-20.10) vers Trident 20.10 ou une version ultérieure, vous pourriez rencontrer une erreur. Pour plus d'informations, consultez "Problèmes connus". Si vous rencontrez l'erreur, vous devez d'abord désactiver le plugin, puis supprimer le plugin, puis installer la version requise de Trident en passant un paramètre de configuration supplémentaire : `docker plugin install netapp/trident-plugin:20.10 --alias netapp --grant-all-permissions config=config.json`

Désinstaller

Effectuez les étapes ci-dessous pour désinstaller Trident pour Docker.

Étapes

1. Supprimez tous les volumes que le plugin a créés.
2. Désactivez le plugin :

```
docker plugin disable netapp:latest
docker plugin ls
ID                  NAME                DESCRIPTION
ENABLED
7067f39a5df5        netapp:latest       nDVP - NetApp Docker Volume
Plugin   false
```

3. Supprimez le plugin :

```
docker plugin rm netapp:latest
```

Travailler avec des volumes

Vous pouvez facilement créer, cloner et supprimer des volumes à l'aide des commandes standard `docker volume` avec le nom du pilote Trident spécifié lorsque cela est nécessaire.

Créer un volume

- Créez un volume avec un driver en utilisant le 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 ne spécifiez aucun "options", les valeurs par défaut du pilote sont utilisées.

- Modifiez la taille de volume par défaut. Consultez l'exemple suivant pour créer un volume de 20 GiB avec un driver :

```
docker volume create -d netapp --name my_vol --opt size=20G
```

ASTUCE Les tailles de volume sont exprimées sous forme de chaînes contenant une valeur entière avec des unités optionnelles (exemple : 10G, 20GB, 3TiB). Si aucune unité n'est spécifiée, la valeur par défaut est G. Les unités de taille peuvent être exprimées soit en puissances de 2 (B, KiB, MiB, GiB, TiB), soit en puissances de 10 (B, KB, MB, GB, TB). Les unités abrégées utilisent les puissances de 2 (G = GiB, T = TiB, …).

Supprimer un volume

- Supprimez le volume comme n'importe quel autre volume Docker :

```
docker volume rm firstVolume
```

IMPORTANT Lors de l'utilisation du `solidfire-san` driver, l'exemple ci-dessus supprime et purge le volume.

Effectuez les étapes ci-dessous pour mettre à niveau Trident pour Docker.

Cloner un volume

Lors de l'utilisation des `ontap-nas`, `ontap-san` et `solidfire-san` pilotes de stockage, Trident peut cloner des volumes. Lors de l'utilisation des `ontap-nas-flexgroup` ou `ontap-nas-economy` pilotes, le clonage 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>
```

- Créez un nouveau volume à partir d'un volume existant. Cela entraînera 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 instantané existant sur un volume. Cela ne créera 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 utilisant Trident **uniquement** s'ils ne comportent aucune partition et si leur système de fichiers est pris en charge par Trident (par exemple : un `ext4`-formaté `/dev/sdc1` ne sera pas accessible via Trident).

Options de volume spécifiques au pilote

Chaque pilote de stockage possède un ensemble d'options différent, que vous pouvez spécifier lors de la création du volume pour personnaliser le résultat. Voir ci-dessous les options qui s'appliquent à votre système de stockage configuré.

L'utilisation de ces options lors de la création d'un volume est simple. Fournissez l'option et la valeur à l'aide de l' `-o`opérateur lors de l'opération CLI. Celles-ci remplacent toutes les valeurs équivalentes du fichier de configuration JSON.

Options de volume ONTAP

Les options de création de volumes pour NFS, iSCSI et FC incluent les suivantes :

Option	Description
`size`	La taille du volume, par défaut à 1 Gio.
`spaceReserve`	Provisionnez le volume en mode fin ou épais, par défaut en mode fin. Les valeurs valides sont `none` (provisionnement fin) et `volume` (provisionnement épais).
`snapshotPolicy`	Cela définira la stratégie de snapshots selon la valeur souhaitée. La valeur par défaut est `none`, ce qui signifie qu'aucun snapshot ne sera automatiquement créé pour le volume. Sauf modification par votre administrateur de stockage, une stratégie nommée "default" existe sur tous les systèmes ONTAP, qui crée et conserve six snapshots horaires, deux quotidiens et deux hebdomadaires. Les données préservées dans un snapshot peuvent être récupérées en accédant au répertoire `.snapshot` dans n'importe quel répertoire du volume.

Option	Description
`snapshotReserve`	Cela définira la réserve de snapshots au pourcentage souhaité. La valeur par défaut est aucune valeur, ce qui signifie que ONTAP sélectionnera la snapshotReserve (généralement 5 %) si vous avez sélectionné une snapshotPolicy, ou 0 % si la snapshotPolicy est none. Vous pouvez définir la valeur par défaut de snapshotReserve dans le fichier de configuration pour tous les backends ONTAP, et vous pouvez l'utiliser comme option de création de volume pour tous les backends ONTAP sauf ontap-nas-economy.
`splitOnClone`	Lors du clonage d'un volume, cela amènera ONTAP à séparer immédiatement le clone de son parent. La valeur par défaut est `false`. Certains cas d'utilisation du clonage de volumes sont mieux servis en séparant immédiatement le clone de son parent lors de la création, car il est peu probable qu'il y ait des opportunités d'efficacité de stockage. Par exemple, le clonage d'une base de données vide peut permettre un gain de temps important mais peu d'économies d'espace de stockage, il est donc préférable de séparer immédiatement le clone.
`encryption`	Activez le chiffrement de volume NetApp (NVE) sur le nouveau volume ; la valeur par défaut est `false`. NVE doit être sous licence et activé sur le cluster pour utiliser cette option. Si NAE est activé sur le backend, tout volume provisionné dans Trident sera activé pour NAE. Pour plus d'informations, consultez : "Comment Trident fonctionne avec NVE et NAE".
`tieringPolicy`	Définit la stratégie de hiérarchisation à utiliser pour le volume. Celle-ci détermine si les données sont déplacées vers le niveau cloud lorsqu'elles deviennent inactives (froides).

Les options supplémentaires suivantes sont pour NFS **uniquement** :

Option	Description
`unixPermissions`	Ceci contrôle l'ensemble des permissions pour le volume lui-même. Par défaut, les permissions seront définies sur `---rwxr-xr-x`, ou en notation numérique 0755, et `root` sera le propriétaire. Le format texte ou numérique fonctionnera.

Option	Description
`snapshotDir`	Définir cette option sur `true` rendra le répertoire `.snapshot` visible pour les clients accédant au volume. La valeur par défaut est `false`, ce qui signifie que la visibilité du répertoire `.snapshot` est désactivée par défaut. Certaines images, par exemple l'image officielle MySQL, ne fonctionnent pas comme prévu lorsque le répertoire `.snapshot` est visible.
`exportPolicy`	Définit les règles d'export à utiliser pour le volume. La valeur par défaut est `default`.
`securityStyle`	Définit le style de sécurité à utiliser pour l'accès au volume. La valeur par défaut est `unix`. Les valeurs valides sont `unix` et `mixed`.

Les options supplémentaires suivantes concernent uniquement iSCSI :

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`	Définir cette option sur `false` désactivera la fonctionnalité d'allocation d'espace du LUN. La valeur par défaut est `true`, ce qui signifie que ONTAP notifie l'hôte lorsque le volume est plein et que le LUN dans le volume ne peut pas accepter d'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 GiB :

```
docker volume create -d netapp --name demo -o size=10G -o
encryption=true
```

- Créer un volume de 100 GiB 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 MiB.

Si la réserve de snapshots n'est pas spécifiée et que la politique de snapshots est `none`, Trident utilise une réserve de snapshots de 0 %.

- Créer un volume sans snapshot policy et sans snapshot reserve :

```
docker volume create -d netapp --name my_vol --opt snapshotPolicy=none
```

- Créez un volume sans stratégie de snapshot et avec une réserve de snapshot personnalisée de 10 % :

```
docker volume create -d netapp --name my_vol --opt snapshotPolicy=none
--opt snapshotReserve=10
```

- Créez un volume avec une stratégie de snapshot et une réserve de snapshot personnalisée de 10 % :

```
docker volume create -d netapp --name my_vol --opt
snapshotPolicy=myPolicy --opt snapshotReserve=10
```

- Créez un volume avec une stratégie de snapshot et acceptez la réserve de snapshot par défaut d'ONTAP (généralement 5 %) :

```
docker volume create -d netapp --name my_vol --opt
snapshotPolicy=myPolicy
```

Options de volume du logiciel Element

Les options du logiciel Element exposent la taille et les politiques de qualité de service (QoS) associées au volume. Lors de la création du volume, la politique QoS qui lui est associée est spécifiée à l'aide de la `-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 minimales, maximales et en rafale associées à un nom dans le fichier de configuration.

Les autres options de création de volumes du logiciel Element incluent les suivantes :

Option	Description
`size`	La taille du volume, par défaut 1 Gio ou entrée de configuration … "defaults" : {"size": "5G"}.
`blocksize`	Utilisez soit 512, soit 4096, par défaut 512 ou l'entrée de configuration DefaultBlockSize.

Exemple

Voir l'exemple de fichier de configuration 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 politique : Bronze, Silver et Gold. Ces noms sont arbitraires.

- Créer un volume Gold de 10 GiB :

```
docker volume create -d solidfire --name sfGold -o type=Gold -o size=10G
```

- Créer un volume Bronze de 100 GiB :

```
docker volume create -d solidfire --name sfBronze -o type=Bronze -o
size=100G
```

Collecter les journaux

Vous pouvez collecter les journaux pour obtenir de l'aide lors du dépannage. La méthode que vous utilisez pour collecter les journaux varie selon la façon dont vous exécutez le plugin Docker.

Collecter les journaux pour le dépannage

Étapes

1. Si vous exécutez Trident en utilisant la méthode de plugin géré recommandée (c'est-à-dire en utilisant `docker plugin` commandes), consultez-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 de journalisation standard devrait vous permettre de diagnostiquer la plupart des problèmes. Si cela ne suffit pas, vous pouvez activer la journalisation de débogage.

2. Pour activer la journalisation de débogage, installez le plugin 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 plugin est déjà installé :

```
docker plugin disable <plugin>
```

```
docker plugin set <plugin> debug=true
```

```
docker plugin enable <plugin>
```

3. Si vous exécutez le fichier binaire directement sur l'hôte, les journaux sont disponibles dans le répertoire `/var/log/netappdvp` de l'hôte. Pour activer la journalisation de débogage, spécifiez `-debug` lors de l'exécution du plugin.

Conseils généraux de dépannage

- Le problème le plus fréquent rencontré par les nouveaux utilisateurs est une erreur de configuration empêchant l'initialisation du plugin. Lorsque cela se produit, vous verrez probablement un message comme celui-ci lorsque vous essayerez d'installer ou d'activer le plugin :

  ```
  Error response from daemon: dial unix /run/docker/plugins/<id>/netapp.sock:
  connect: no such file or directory
  ```

  Cela signifie que le plugin n'a pas pu démarrer. Heureusement, le plugin a été conçu avec une capacité de journalisation complète qui devrait vous aider à diagnostiquer la plupart des problèmes que vous êtes susceptible de rencontrer.

- En cas de problème lors du montage d'un PV dans un conteneur, assurez-vous que `rpcbind`est installé et en cours d'exécution. Utilisez le gestionnaire de paquets 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 une `systemctl status rpcbind` ou une commande équivalente.

Gérer plusieurs instances Trident

Plusieurs instances de Trident sont nécessaires lorsque vous souhaitez disposer simultanément de plusieurs configurations de stockage. La clé pour utiliser plusieurs instances est de leur donner des noms différents en utilisant l' `--alias`option avec le plugin conteneurisé, ou l' `--volume-driver`option lors de l'instanciation de Trident sur l'hôte.

Étapes pour le plugin 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 alias différent 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 en spécifiant l'alias comme nom du 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 la version traditionnelle (1.12 ou antérieure)

1. Lancez le plugin avec une configuration NFS en utilisant un ID de pilote personnalisé :

   ```
   sudo trident --volume-driver=netapp-nas --config=/path/to/config
   -nfs.json
   ```

2. Lancez le plugin avec une configuration iSCSI en utilisant un ID de pilote personnalisé :

   ```
   sudo trident --volume-driver=netapp-san --config=/path/to/config
   -iscsi.json
   ```

3. Provisionnez des 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 globales

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 optionnel pour les noms de volumes. Par défaut : `netappdvp_`.	`staging_`
`limitVolumeSize`	Restriction optionnelle sur la taille des volumes. Par défaut : "" (non appliquée)	`10g`

ASTUCE

Ne pas utiliser `storagePrefix` (y compris la valeur par défaut) pour les backends Element. Par défaut, le `solidfire-san` driver ignorera ce paramètre et n'utilisera pas de préfixe. NetApp recommande d'utiliser soit un tenantID spécifique pour le mappage des volumes Docker, soit l'attribut data qui est renseigné avec la version de Docker, les informations du driver et le nom brut provenant de Docker dans les cas où un traitement du nom aurait pu être utilisé.

Des options par défaut sont disponibles pour éviter de devoir les spécifier à chaque volume que vous créez. L' `size`option est disponible pour tous les types de contrôleurs. Voir la section de configuration ONTAP pour un exemple de définition de la taille de volume par défaut.

Option	Description	Exemple
`size`	Taille par défaut facultative pour les nouveaux volumes. Par défaut : `1G`	`10G`

Configuration ONTAP

En plus des valeurs de configuration globales ci-dessus, lors de l'utilisation de ONTAP, les options de niveau supérieur suivantes sont disponibles.

Option	Description	Exemple
`managementLIF`	Adresse IP de la LIF de gestion ONTAP. Vous pouvez spécifier un nom de domaine complet (FQDN).	`10.0.0.1`

Option	Description	Exemple
`dataLIF`	Adresse IP de l'interface logique de protocole (LIF). **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 complet (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. **ONTAP SAN drivers** : Ne spécifiez pas pour iSCSI ou FC. Trident utilise "Mappage LUN sélectif ONTAP" pour découvrir les LIF iSCSI ou FC nécessaires à l'établissement d'une session multi-chemin. Un avertissement est généré si `dataLIF` est explicitement défini.	`10.0.0.2`
`svm`	Machine virtuelle de stockage à utiliser (obligatoire, si l'interface LIF de gestion est une interface LIF de cluster)	`svm_nfs`
`username`	Nom d'utilisateur pour se connecter au périphérique de stockage	`vsadmin`
`password`	Mot de passe pour se connecter au périphérique de stockage	`secret`
`aggregate`	Agrégat pour le provisionnement (facultatif ; s'il est défini, il doit être attribué à la SVM). Pour le `ontap-nas-flexgroup` driver, cette option est ignorée. Tous les agrégats affectés au SVM sont utilisés pour provisionner un FlexGroup volume.	`aggr1`
`limitAggregateUsage`	Facultatif, l'approvisionnement échoue si l'utilisation dépasse ce pourcentage	`75%`

Option	Description	Exemple
`nfsMountOptions`	Contrôle précis des options de montage NFS ; par défaut : « -o nfsvers=3 ». **Disponible uniquement pour les `ontap-nas` et `ontap-nas-economy` pilotes**. "Consultez les informations de configuration de l'hôte NFS 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` driver**.	`netappdvp`
`limitVolumeSize`	Taille maximale de volume pouvant être demandée.	`300g`
`qtreesPerFlexvol`	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` driver, cette option permet de personnaliser le nombre maximal de qtrees par FlexVol**.	`300`
`sanType`	**Prise en charge pour `ontap-san` le pilote uniquement.** Utilisez pour sélectionner `iscsi` pour iSCSI, `nvme` pour NVMe/TCP ou `fcp` pour SCSI sur Fibre Channel (FC).	`iscsi` si vide
`limitVolumePoolSize`	**Pris en charge pour `ontap-san-economy` et `ontap-san-economy` pilotes uniquement.** Limite les tailles de FlexVol dans les pilotes ONTAP ontap-nas-economy et ontap-SAN-economy.	`300g`

Des 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` (thin provisioned) ou `volume` (thick)	`none`

Option	Description	Exemple
`snapshotPolicy`	Stratégie d'instantané à utiliser, la valeur par défaut est `none`	`none`
`snapshotReserve`	Pourcentage de réserve d'instantané, la valeur par défaut est « » pour accepter la valeur par défaut ONTAP	`10`
`splitOnClone`	Séparer un clone de son parent lors de sa création, par défaut à `false`	`false`
`encryption`	Active NetApp Volume Encryption (NVE) sur le nouveau volume ; la valeur par défaut est `false`. NVE doit être sous licence et activé sur le cluster pour utiliser cette option. Si NAE est activé sur le backend, tout volume provisionné dans Trident sera activé pour NAE. Pour plus d'informations, consultez : "Comment Trident fonctionne avec NVE et NAE".	true
`unixPermissions`	Option NAS pour les volumes NFS provisionnés, par défaut `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`	Option NAS pour la règles d'export à utiliser, par défaut `default`	`default`
`securityStyle`	Option NAS pour l'accès au volume NFS provisionné. NFS prend en charge `mixed` et `unix` styles de sécurité. La valeur par défaut est `unix`.	`unix`
`fileSystemType`	Option SAN permettant de sélectionner le type de système de fichiers, par défaut `ext4`	`xfs`
`tieringPolicy`	Politique de hiérarchisation à utiliser, la valeur par défaut est `none`.	`none`

Option	Description	Exemple
`skipRecovery Queue`	Lors de la suppression d'un volume, ignorez la file d'attente de récupération dans le stockage et supprimez le volume immédiatement.	``

Options de mise à l'échelle

Les `ontap-nas` et `ontap-san` pilotes créent un ONTAP FlexVol pour chaque volume Docker. ONTAP prend en charge jusqu'à 1 000 FlexVols par nœud de cluster avec un maximum de 12 000 volumes FlexVol par cluster. Si vos besoins en volumes Docker respectent cette limite, le `ontap-nas` pilote est la solution NAS privilégiée grâce aux fonctionnalités supplémentaires offertes par les FlexVols, telles que les snapshots et le clonage granulaires au niveau du volume Docker.

Si vous avez besoin de plus de volumes Docker que ne le permettent les limites FlexVol, choisissez le `ontap-nas-economy` ou le `ontap-san-economy` pilote.

Le `ontap-nas-economy` pilote crée des volumes Docker sous forme de Qtrees ONTAP au sein d'un pool de volumes FlexVol gérés automatiquement. Les Qtrees offrent une évolutivité bien 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 ou le clonage granulaires de volumes Docker.

REMARQUE	Le `ontap-nas-economy`driver n'est actuellement pas pris en charge dans Docker Swarm, car Docker Swarm n'orchestré pas la création de volumes sur plusieurs nœuds.

Le `ontap-san-economy` pilote crée des volumes Docker sous forme de LUN ONTAP au sein d'un pool partagé de FlexVol volumes gérés automatiquement. Ainsi, chaque FlexVol n'est pas limité à un seul LUN et cela offre une meilleure scalabilité pour les charges de travail SAN. Selon la baie de stockage, ONTAP prend en charge jusqu'à 16384 LUN par cluster. Parce que les volumes sont des LUN sous-jacents, ce pilote prend en charge les snapshots et le clonage granulaires au niveau des volumes Docker.

Choisissez le `ontap-nas-flexgroup` driver pour augmenter le parallélisme sur un volume unique pouvant atteindre la plage du pétaoctet avec des milliards de fichiers. Quelques cas d'utilisation idéaux pour FlexGroups incluent l'IA/ML/DL, le big data et l'analytique, les compilations logicielles, le streaming, les référentiels de fichiers, etc. Trident utilise tous les agrégats affectés à une SVM lors du provisionnement d'un volume FlexGroup. La prise en charge de FlexGroup dans Trident comporte également les considérations suivantes :

- Nécessite ONTAP version 9.2 ou supérieure.
- À l'heure actuelle, FlexGroups ne prend en charge que NFS v3.
- Il est recommandé d'activer les identifiants NFSv3 64 bits pour la SVM.
- La taille minimale recommandée d'un membre/volume FlexGroup est de 100 GiB.
- 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 "NetApp FlexGroup volume Guide des meilleures pratiques et de mise en œuvre".

Pour bénéficier de fonctionnalités avancées et d'une grande capacité dans le même environnement, vous pouvez exécuter plusieurs instances du plugin Docker Volume, l'une utilisant `ontap-nas` et l'autre utilisant `ontap-nas-economy`.

Rôle ONTAP personnalisé pour Trident

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

Reportez-vous à "Générateur de rôles personnalisés Trident" pour plus d'informations sur la création de rôles personnalisés Trident.

Utilisation de l'interface de ligne de commande ONTAP

1. Créez un nouveau 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. Associez le rôle à l'utilisateur :

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

Utilisation de System Manager

Effectuez les étapes suivantes dans ONTAP System Manager :

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

      (Ou) Pour créer un rôle personnalisé au niveau de la SVM, sélectionnez **Storage > Storage VMs > `required SVM`> Paramètres > Utilisateurs et rôles**.

   b. Sélectionnez l'icône flèche (→) à côté de **Users and Roles**.
   c. Sélectionnez **+Add** sous **Roles**.
   d. Définissez les règles du rôle et cliquez sur **Save**.
2. **Associez le rôle à l'utilisateur Trident** : + Effectuez les étapes suivantes sur la page **Utilisateurs et rôles** :
   a. Sélectionnez l'icône Ajouter **+** sous **Users**.
   b. Sélectionnez le nom d'utilisateur requis, puis sélectionnez un rôle dans le menu déroulant pour **Rôle**.
   c. Cliquez sur **Enregistrer**.

Reportez-vous aux pages suivantes pour plus d'informations :

- "Rôles personnalisés pour l'administration d'ONTAP" ou "Définir des rôles personnalisés"
- "Travailler avec les rôles et les utilisateurs"

Exemples de fichiers de configuration ONTAP

Exemple NFS pour <code>ontap-nas</code> driver

```
{
  "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 <code>ontap-nas-flexgroup</code> driver

```
{
  "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 <code>ontap-nas-economy</code> driver

```
{
  "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 <code>ontap-san</code> driver

```
{
  "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 <code>ontap-san-economy</code> driver

```
{
  "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 <code>ontap-san</code> driver

```
{
  "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 <code>ontap-san</code> driver

```
{
  "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 du logiciel Element

En plus des valeurs de configuration globales, lors de l'utilisation du logiciel Element (NetApp HCI/SolidFire), 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`	Adresse IP et port iSCSI	10.0.0.7:3260
`TenantName`	SolidFireF Tenant à utiliser (créé s'il n'est pas trouvé)	`docker`

Option	Description	Exemple
`InitiatorIFace`	Spécifiez l'interface lorsque vous limitez le trafic iSCSI à une interface non par défaut	`default`
`Types`	Spécifications QoS	Voir l'exemple ci-dessous
`LegacyNamePrefix`	Préfixe pour les installations Trident mises à niveau. Si vous utilisiez une version de Trident antérieure à 1.3.2 et que vous effectuez une mise à niveau avec des volumes existants, vous devrez définir cette valeur pour accéder à vos anciens volumes qui étaient mappés via la méthode volume-name.	`netappdvp-`

Le `solidfire-san` pilote ne prend pas en charge Docker Swarm.

Exemple de fichier de configuration Element software

```
{
  "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 limitations connus

Trouvez des informations sur les problèmes connus et les limitations lors de l'utilisation de Trident avec Docker.

La mise à niveau du plugin Trident Docker Volume vers la version 20.10 et ultérieures à partir de versions plus anciennes entraîne un échec de mise à niveau avec l'erreur « no such file or directory ».

Solution de contournement

1. Désactivez le plugin.

```
docker plugin disable -f netapp:latest
```

2. Supprimez le plugin.

```
docker plugin rm -f netapp:latest
```

3. Réinstallez le plugin en fournissant le paramètre supplémentaire `config`.

```
docker plugin install netapp/trident-plugin:20.10 --alias netapp --grant
-all-permissions config=config.json
```

Les noms de volumes doivent comporter au minimum 2 caractères.

REMARQUE	Il s'agit d'une limitation du client Docker. Le client interprétera un nom d'un seul caractère comme étant un chemin d'accès Windows. "Voir le bug 25773".

Docker Swarm présente certains comportements qui empêchent Trident de le prendre en charge avec chaque combinaison de stockage et de pilote.

- Docker Swarm utilise actuellement le nom du volume au lieu de l'ID du volume comme identifiant unique de volume.
- Les requêtes de volume sont envoyées simultanément à chaque nœud dans un cluster Swarm.
- Les plugins de volume (y compris Trident) doivent s'exécuter indépendamment sur chaque nœud d'un cluster Swarm. En raison de la façon dont ONTAP fonctionne et de la manière dont les `ontap-nas` et `ontap-san` pilotes fonctionnent, ce sont les seuls qui peuvent opérer dans ces limites.

Les autres pilotes sont sujets à des problèmes tels que des conditions de concurrence qui peuvent entraîner la création d'un grand nombre de volumes pour une seule requête sans "gagnant" clair ; par exemple, Element possède une fonctionnalité qui permet à des volumes d'avoir le même nom mais des identifiants différents.

NetApp a fourni des commentaires à l'équipe Docker, mais n'a aucune indication de recours futur.

Si un FlexGroup est en cours de provisionnement, ONTAP ne provisionne pas un second FlexGroup si le second FlexGroup a un ou plusieurs agrégats en commun avec le FlexGroup en cours de provisionnement.

Meilleures pratiques et recommandations

Déploiement

Utilisez les recommandations énumérées ici lors du déploiement de Trident.

Déployer dans un espace de noms dédié

"Espaces de noms" assurent une séparation administrative entre différentes applications et constituent une barrière au partage des ressources. Par exemple, un PVC d'un espace de noms ne peut pas être consommé depuis un autre. Trident fournit des ressources PV à tous les espaces de noms du cluster Kubernetes et exploite ainsi un compte de service disposant de privilèges élevés.

De plus, l'accès au pod Trident pourrait 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 n'ont pas la possibilité d'accéder aux définitions d'objets Trident ou aux pods eux-mêmes.

Utilisez des quotas et des limites de plage pour contrôler la consommation de stockage

Kubernetes possède deux fonctionnalités qui, combinées, offrent un mécanisme puissant pour limiter la consommation de ressources par les applications. Le "mécanisme de quota de stockage" permet à l'administrateur de mettre en œuvre des limites de consommation de capacité et de nombre d'objets, globales et spécifiques à chaque classe de stockage, sur une base par espace de noms. De plus, l'utilisation d'un "limite de portée" garantit que les requêtes PVC respectent à la fois une valeur minimale et maximale avant que la demande ne soit transmise au provisionneur.

Ces valeurs sont définies pour chaque espace de noms, ce qui signifie que chaque espace de noms doit avoir des valeurs définies qui correspondent à ses besoins en ressources. Voir ici pour des informations à propos de "comment tirer parti des quotas".

Configuration du stockage

Chaque plateforme de stockage dans le portefeuille NetApp possède des capacités uniques qui bénéficient aux applications, conteneurisées ou non.

Présentation de la plateforme

Trident fonctionne avec ONTAP et Element. Il n'existe pas de plateforme qui soit mieux adaptée à toutes les applications et à tous les scénarios qu'une autre ; cependant, les besoins de l'application et de l'équipe qui administre le dispositif doivent être pris en compte lors du choix d'une plateforme.

Il est recommandé de suivre les bonnes pratiques de base pour le système d'exploitation hôte avec le protocole que vous utilisez. Vous pouvez également envisager d'intégrer les bonnes pratiques applicatives, lorsqu'elles sont disponibles, avec le backend, la classe de stockage et les paramètres PVC afin d'optimiser le stockage pour des applications spécifiques.

Meilleures pratiques ONTAP et Cloud Volumes ONTAP

Découvrez les meilleures pratiques pour configurer ONTAP et Cloud Volumes ONTAP pour Trident.

Les recommandations suivantes constituent des lignes directrices pour la configuration ONTAP pour les charges de travail conteneurisées, qui consomment des volumes provisionnés dynamiquement par Trident. Chacune d'elles doit être examinée et évaluée en fonction de sa pertinence pour votre environnement.

Utilisez des SVM dédiés à Trident

Les machines virtuelles de stockage (SVM) assurent l'isolation et la séparation administrative entre les locataires sur un système ONTAP. L'attribution d'une SVM aux applications permet la délégation de privilèges et l'application des meilleures pratiques pour limiter la consommation de ressources.

Plusieurs options sont disponibles pour la gestion du SVM :

- Fournissez l'interface de gestion du cluster dans la configuration du backend, ainsi que les informations d'identification appropriées, et spécifiez le nom du SVM.
- Créez une interface de gestion dédiée pour la SVM en utilisant ONTAP System Manager ou la CLI.
- Partagez le rôle de gestion avec une interface de données NFS.

Dans chaque cas, l'interface doit être dans le DNS, et le nom DNS doit être utilisé lors de la configuration de Trident. Cela aide à faciliter certains scénarios de reprise après sinistre, par exemple, SVM-DR sans l'utilisation de la conservation de l'identité réseau.

Il n'y a pas de préférence entre une LIF de gestion dédiée ou partagée pour la SVM, toutefois, vous devez vous assurer que vos politiques de sécurité réseau sont compatibles avec l'approche que vous choisissez. Dans tous les cas, la LIF de gestion doit être accessible via DNS pour offrir une flexibilité maximale si "SVM-DR" est utilisée conjointement avec Trident.

Limiter le nombre maximal de volumes

Les systèmes de stockage ONTAP ont un nombre maximal de volumes, qui varie selon la version du logiciel et la plateforme matérielle. Consultez "NetApp Hardware Universe" pour votre plateforme spécifique et votre version d'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 toutes les demandes de stockage.

Les pilotes de Trident `ontap-nas` et `ontap-san` provisionnent un FlexVolume pour chaque volume persistant (PV) Kubernetes créé. Le pilote `ontap-nas-economy` crée environ un FlexVolume pour 200 PV (configurable entre 50 et 300). Le pilote `ontap-san-economy` crée environ un FlexVolume pour 100 PV (configurable entre 50 et 200). Pour éviter que Trident ne consomme tous les volumes disponibles sur le système de stockage, vous devez définir une limite sur le SVM. Vous pouvez le faire depuis la ligne de commandes :

```
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
- Le nombre de volumes persistants attendus à être consommés par les applications Kubernetes

La `max-volumes` valeur correspond au total des volumes provisionnés sur l'ensemble des nœuds du cluster ONTAP, et non sur un nœud ONTAP individuel. Par conséquent, il se peut qu'un nœud de cluster ONTAP ait beaucoup plus ou beaucoup moins de volumes Trident provisionnés qu'un autre nœud.

Par exemple, un cluster ONTAP à deux nœuds peut héberger un maximum de 2 000 volumes FlexVol. Fixer le nombre maximal de volumes à 1 250 semble très raisonnable. Cependant, si seuls les "agrégats" volumes d'un nœud sont affectés à la SVM, ou si les agrégats affectés d'un nœud ne peuvent pas être provisionnés (par exemple, en raison de la capacité), alors l'autre nœud devient la cible de tous les volumes provisionnés par Trident. Cela signifie que la limite de volumes de ce nœud peut être atteinte avant que la valeur `max-volumes` ne soit atteinte, ce qui impacte à la fois Trident et les autres opérations de volumes utilisant ce nœud. **Vous pouvez éviter cette situation en vous assurant que les agrégats de chaque nœud du cluster sont affectés à la SVM utilisée par Trident en nombre égal.**

Cloner un volume

NetApp Trident prend en charge le clonage de volumes lors de l'utilisation des `ontap-nas`, `ontap-san` et `solidfire-san` pilotes de stockage. Lors de l'utilisation des `ontap-nas-flexgroup` ou `ontap-nas-economy` pilotes, le clonage 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 un StorageClass différent. Effectuez les opérations de clonage au sein du même StorageClass pour garantir la compatibilité et éviter tout comportement inattendu.

Limiter la taille maximale des volumes créés par Trident

Pour configurer la taille maximale des volumes pouvant être créés par Trident, utilisez le `limitVolumeSize` paramètre dans votre `backend.json` définition.

En plus de contrôler la taille du volume au niveau de la baie de stockage, vous devriez également tirer parti des capacités de Kubernetes.

Limiter la taille maximale des FlexVols créés par Trident

Pour configurer la taille maximale pour les FlexVols utilisées 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 CHAP bidirectionnel

Vous pouvez spécifier les noms d'utilisateur et mots de passe de l'initiateur et de la cible CHAP dans la définition de votre backend et demander à Trident d'activer CHAP sur la SVM. En utilisant le paramètre `useCHAP` dans la configuration de votre backend, Trident authentifie les connexions iSCSI pour les backends ONTAP avec CHAP.

Créer et utiliser une règle QoS SVM

L'utilisation d'une politique QoS ONTAP, appliquée à la SVM, limite le nombre d'IOPS consommables par les volumes provisionnés Trident. Cela permet d' "empêcher un tyran"éviter qu'un conteneur défaillant ou incontrôlé n'affecte les charges de travail en dehors de la SVM Trident.

Vous pouvez créer une politique QoS pour la SVM en quelques étapes. Consultez la documentation de votre version de ONTAP pour obtenir les informations les plus précises. L'exemple ci-dessous crée une politique 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>
```

De plus, si votre version d'ONTAP le permet, vous pouvez envisager d'utiliser un minimum de QoS pour garantir un certain débit aux charges de travail conteneurisées. La QoS adaptative n'est pas compatible avec une politique au niveau SVM.

Le nombre d'IOPS dédiés aux charges de travail conteneurisées dépend de nombreux aspects. Parmi ceux-ci, on trouve notamment :

- Autres charges de travail utilisant la baie de stockage. Si d'autres charges de travail, non liées au déploiement Kubernetes, utilisent les ressources de stockage, il convient de veiller à ce qu'elles ne soient pas accidentellement affectées négativement.
- Charges de travail prévues exécutées dans des conteneurs. Si des charges de travail ayant des exigences élevées en IOPS s'exécutent dans des conteneurs, une politique de QoS faible entraîne une mauvaise expérience.

Il est important de noter qu'une politique QoS définie au niveau de la SVM implique que tous les volumes provisionnés sur la SVM partagent le même pool d'IOPS. Si une application conteneurisée, ou un petit nombre d'applications conteneurisées, a des besoins élevés en IOPS, elle risque de pénaliser les autres charges de travail conteneurisées. Si tel est le cas, vous pouvez envisager d'utiliser une automatisation externe pour attribuer des politiques QoS par volume.

IMPORTANT	Vous ne devez attribuer le groupe de stratégie QoS au SVM **que** si votre version d'ONTAP est antérieure à 9.8.

Créer des groupes de règles QoS pour Trident

La qualité de service (QoS) garantit que les performances des charges de travail critiques ne sont pas dégradées par des charges de travail concurrentes. Les groupes de règles QoS d'ONTAP offrent 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, consultez "Garantir le débit avec QoS". Vous pouvez spécifier des groupes de règles QoS dans le backend ou dans un pool de stockage, et ils sont appliqués à chaque volume créé dans ce pool ou backend.

ONTAP propose deux types de groupes de stratégies QoS : traditionnels et adaptatifs. Les groupes de stratégies traditionnels offrent un débit maximal (ou minimal, dans les versions plus récentes) fixe en IOPS. La QoS adaptative ajuste automatiquement le débit à la taille de la charge de travail, en maintenant le ratio IOPS à To|Go à mesure que la taille de la charge de travail évolue. Cela représente un avantage considérable lorsque vous gérez des centaines ou des milliers de charges de travail dans un déploiement de grande envergure.

Tenez compte des points suivants lors de la création de groupes de règles QoS :

- Vous devez définir la `qosPolicy` clé dans le bloc `defaults` de la configuration du backend. Voir l'exemple de configuration du backend 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 groupes de stratégies par volume, afin que chaque volume bénéficie de l'ensemble du débit spécifié par le groupe de stratégies. Les groupes de stratégies partagés ne sont pas pris en charge.

Pour plus d'informations sur les groupes de politiques QoS, consultez "Référence des commandes ONTAP".

Limiter l'accès aux ressources de stockage aux membres du cluster Kubernetes

Limiter l'accès aux volumes NFS, aux LUN iSCSI et aux LUN FC créés par Trident est un élément essentiel de la posture de sécurité de votre déploiement Kubernetes. Cela empêche les hôtes qui ne font pas partie du cluster Kubernetes d'accéder aux volumes et de modifier potentiellement les données de manière inattendue.

Il est important de comprendre que les espaces de noms constituent la limite logique des ressources dans Kubernetes. On suppose que les ressources dans le même espace de noms peuvent être partagées, cependant, il est important de noter qu'il n'existe aucune capacité inter-espaces de noms. Cela signifie que même si les PV sont des objets globaux, lorsqu'ils sont liés à un PVC, ils ne sont accessibles que par les pods qui sont dans le même espace de noms. **Il est essentiel de veiller à utiliser les espaces de noms pour assurer la séparation lorsque cela est approprié.**

La principale préoccupation de la plupart des organisations concernant la sécurité des données dans un contexte Kubernetes est qu'un processus exécuté dans un conteneur puisse accéder à un stockage monté sur l'hôte, mais qui n'est pas destiné au conteneur. "Espaces de noms" sont conçus pour empêcher ce type de compromission. Il existe cependant une exception : les conteneurs privilégiés.

Un conteneur privilégié est un conteneur exécuté avec des permissions au niveau de l'hôte nettement supérieures à la normale. Celles-ci ne sont pas refusées par défaut, alors assurez-vous de désactiver la capacité en utilisant "politiques de sécurité des pods".

Pour les volumes nécessitant un accès depuis Kubernetes et des hôtes externes, le stockage doit être géré de manière traditionnelle, avec la création du PV par l'administrateur et non géré par Trident. Cela garantit que le

volume de stockage n'est détruit que lorsque Kubernetes et les hôtes externes sont déconnectés et n'utilisent plus le volume. De plus, une règle d'export peut être appliquée, ce qui permet l'accès depuis les nœuds du cluster Kubernetes et les serveurs cibles situés en dehors du cluster Kubernetes.

Pour les déploiements comportant des nœuds d'infrastructure dédiés (par exemple, OpenShift) ou d'autres nœuds incapables de planifier des applications utilisateur, des règles d'export distinctes doivent être utilisées afin de limiter davantage l'accès aux ressources de stockage. Cela inclut la création d'une règle d'export pour les services déployés sur ces nœuds d'infrastructure (par exemple, les services de métriques et de journalisation OpenShift), ainsi que pour les applications standard déployées sur des nœuds non dédiés à l'infrastructure.

Utilisez une règle d'export

Vous devez vous assurer qu'une règle d'export existe pour chaque backend, autorisant uniquement l'accès aux nœuds présents dans le cluster Kubernetes. Trident peut créer et gérer automatiquement des règles d'export. Ainsi, Trident limite l'accès aux volumes qu'il provisionne aux nœuds du cluster Kubernetes et simplifie l'ajout et la suppression de nœuds.

Vous pouvez également créer une règle d'export manuellement et la remplir avec une ou plusieurs règles d'export qui traitent chaque demande d'accès au nœud :

- Utilisez la `vserver export-policy create` commande ONTAP CLI pour créer la règle d'export.
- Ajoutez des règles à la règles d'export à l'aide de la commande CLI `vserver export-policy rule create` ONTAP.

L'exécution de ces commandes vous permet de restreindre quels nœuds Kubernetes ont accès aux données.

Désactiver `showmount` pour l'application SVM

La fonctionnalité `showmount` permet à un client NFS d'interroger le SVM pour obtenir une liste des exports NFS disponibles. Un pod déployé sur le cluster Kubernetes peut exécuter la commande `showmount -e` contre le SVM et recevoir une liste des montages disponibles, y compris ceux auxquels il n'a pas accès. Bien que cela, en soi, ne constitue pas une compromission de la sécurité, cela fournit des informations inutiles susceptibles d'aider un utilisateur non autorisé à se connecter à un export NFS.

Vous devez désactiver `showmount` en utilisant la commande ONTAP CLI au niveau SVM :

```
vserver nfs modify -vserver <svm_name> -showmount disabled
```

SolidFire meilleures pratiques

Découvrez les meilleures pratiques pour configurer le stockage SolidFire pour Trident.

Créer un compte SolidFire

Chaque compte SolidFire représente un propriétaire de volume unique et reçoit ses propres identifiants Challenge-Handshake Authentication Protocol (CHAP). Vous pouvez accéder aux volumes associés à un compte soit en utilisant le nom du compte et les identifiants CHAP correspondants, soit via un groupe d'accès aux volumes. Un compte peut avoir jusqu'à deux mille volumes qui lui sont attribués, mais un volume ne peut appartenir qu'à un seul compte.

Créer une politique QoS

Utilisez les politiques de qualité de service (QoS) SolidFire si vous souhaitez créer et enregistrer un paramètre de qualité de service standardisé qui peut être appliqué à de nombreux volumes.

Vous pouvez configurer les paramètres QoS volume par volume. Les performances de chaque volume peuvent être garanties en définissant trois paramètres configurables qui définissent la QoS : Min IOPS, Max IOPS et Burst IOPS.

Voici les valeurs minimales, maximales et en rafale d'IOPS possibles pour la taille de bloc 4Kb.

Paramètre IOPS	Définition	Valeur min.	valeur par défaut	Valeur maximale (4Kb)
IOPS minimales	Le niveau de performance garanti pour un volume.	50	50	15000
IOPS max	Les performances ne dépasseront pas cette limite.	50	15000	200 000
IOPS en rafale	Nombre maximal d'IOPS autorisés dans un scénario de rafale courte.	50	15000	200 000

REMARQUE	Bien que les valeurs Max IOPS et Burst IOPS puissent être fixées 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 les groupes d'accès aux volumes (VAG). CHAP utilise le protocole CHAP pour authentifier l'hôte auprès du backend. Les groupes d'accès aux volumes contrôlent l'accès aux volumes qu'ils provisionnent. NetApp recommande d'utiliser CHAP pour l'authentification, car c'est plus simple et il n'y a pas de limites de mise à l'échelle.

REMARQUE	Trident avec le provisionneur CSI amélioré prend en charge l'authentification CHAP. Les VAG doivent être utilisés uniquement en mode de fonctionnement traditionnel non-CSI.

L'authentification CHAP (vérification que l'initiateur est bien l'utilisateur du volume) est prise en charge uniquement avec le contrôle d'accès basé sur les comptes. Si vous utilisez CHAP pour l'authentification, deux options sont disponibles : CHAP unidirectionnel et CHAP bidirectionnel. CHAP unidirectionnel authentifie l'accès au volume à l'aide du nom de compte SolidFire et du secret de l'initiateur. L'option CHAP bidirectionnel offre la méthode d'authentification la plus sécurisée, car le volume authentifie l'hôte via le nom de compte et le

secret de l'initiateur, puis l'hôte authentifie le volume via le nom de compte et le secret de la cible.

Toutefois, si CHAP ne peut être activé et que des groupes d'accès aux volumes (VAG) 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, le contrôle d'accès basé sur le compte est utilisé. Si l'initiateur iSCSI n'est pas configuré pour utiliser l'authentification CHAP, alors le contrôle d'accès du Volume Access Group est utilisé.

Où trouver plus d'informations ?

Certains documents de bonnes pratiques sont listés ci-dessous. Recherchez dans "Bibliothèque NetApp" les versions les plus récentes.

ONTAP

- "Guide des meilleures pratiques et de mise en œuvre NFS"
- "Administration SAN" (pour iSCSI)
- "Configuration iSCSI Express pour RHEL"

Logiciel Element

- "Configuration de SolidFire pour Linux"

NetApp HCI

- "NetApp HCI Prérequis de déploiement"
- "Accédez au NetApp Deployment Engine"

Informations sur les meilleures pratiques d'application

- "Meilleures pratiques pour MySQL sur ONTAP"
- "Meilleures pratiques pour MySQL sur SolidFire"
- "NetApp SolidFire et Cassandra"
- "Meilleures pratiques Oracle sur SolidFire"
- "Meilleures pratiques PostgreSQL sur SolidFire"

Toutes les applications ne disposent pas de directives spécifiques, il est important de travailler avec votre NetApp équipe et d'utiliser le "Bibliothèque NetApp" pour trouver la documentation la plus récente.

Intégrer Trident

Pour intégrer Trident, les éléments de conception et d'architecture suivants doivent être intégrés : sélection et déploiement des pilotes, conception de la classe de stockage, conception du pool virtuel, impacts de la revendication de volume persistant (PVC) sur l'approvisionnement du stockage, opérations sur les volumes et déploiement des services OpenShift utilisant Trident.

Sélection et déploiement du pilote

Sélectionnez et déployez un pilote backend pour votre système de stockage.

Pilotes backend ONTAP

Les pilotes backend ONTAP se distinguent par le protocole utilisé et la manière dont les volumes sont provisionnés sur le système de stockage. Il convient donc d'examiner attentivement le pilote à déployer.

À un niveau supérieur, si votre application comporte des composants nécessitant un stockage partagé (plusieurs pods accédant au même PVC), les pilotes NAS seraient le choix par défaut, tandis que les pilotes iSCSI basés sur les blocs répondent aux besoins d'un stockage non partagé. Choisissez le protocole en fonction des exigences de l'application et du niveau de confort des équipes de stockage et d'infrastructure. De manière générale, il existe peu de différences entre eux pour la plupart des applications, donc la décision est souvent basée sur la nécessité ou non d'un stockage partagé (lorsque plus d'un pod aura besoin d'un accès simultané).

Les pilotes backend ONTAP disponibles sont :

- `ontap-nas` : Chaque PV provisionné est un ONTAP FlexVolume.
- `ontap-nas-economy`: Chaque PV provisionné est un qtree, avec un nombre configurable de qtrees par FlexVolume (la valeur par défaut est 200).
- `ontap-nas-flexgroup`: Chaque PV est provisionné en tant que ONTAP FlexGroup, et tous les agrégats affectés à une SVM sont utilisés.
- `ontap-san` : Chaque PV provisionné est un LUN au sein de son propre FlexVolume.
- `ontap-san-economy`: Chaque PV provisionné est un LUN, avec un nombre configurable de LUN par FlexVolume (100 par défaut).

Le choix entre les trois pilotes NAS a des répercussions sur les fonctionnalités qui sont mises à la disposition de l'application.

Notez que, dans les tableaux ci-dessous, toutes les fonctionnalités ne sont pas accessibles via Trident. Certaines doivent être appliquées par l'administrateur de stockage après le provisionnement si cette fonctionnalité est souhaitée. Les notes de bas de page en exposant distinguent la fonctionnalité par fonctionnalité et par pilote.

Pilotes ONTAP NAS	Snapshots	Clones	Politiques d'exportation dynamiques	Multi-attach	QoS	Redimensionner	Réplication
`ontap-nas`	Oui	Oui	Oui	Oui	Oui	Oui	Oui
`ontap-nas-economy`	NO [3]	NO [3]	Oui	Oui	NO [3]	Oui	NO [3]
`ontap-nas-flexgroup`	Oui	NON	Oui	Oui	Oui	Oui	Oui

Trident propose 2 pilotes SAN pour ONTAP, dont les capacités sont présentées ci-dessous.

Pilotes ONTAP SAN	Snapshots	Clones	Multi-attach	CHAP bidirectionnel	QoS	Redimensionner	Réplication
`ontap-san`	Oui	Oui	Oui	Oui	Oui	Oui	Oui
`ontap-san-economy`	Oui	Oui	Oui	Oui	NO [3]	Oui	NO [3]

> Note de bas de page pour les tableaux ci-dessus : Yes [1] : Non géré par Trident Yes [2] : Géré par Trident, mais pas au niveau PV granulaire NO [3] : Non géré par Trident et pas au niveau PV granulaire Yes [4] : Pris en charge pour les volumes raw-block Yes [5] : Pris en charge par Trident

Les fonctionnalités qui ne sont pas granulaires au niveau du PV sont appliquées à l'ensemble du FlexVolume et tous les PV (c'est-à-dire les qtrees ou les LUNs dans des FlexVols partagés) partageront un calendrier commun.

Comme nous pouvons le voir dans les tableaux ci-dessus, une grande partie de la fonctionnalité entre le `ontap-nas` et le `ontap-nas-economy` est identique. Cependant, parce que le `ontap-nas-economy` pilote limite la capacité de contrôler la planification à la granularité par PV, cela peut affecter en particulier votre planification de reprise après sinistre et de sauvegarde. Pour les équipes de développement qui souhaitent exploiter la fonctionnalité de clonage de PVC sur le stockage ONTAP, cela n'est possible qu'en utilisant les `ontap-nas`, `ontap-san` ou `ontap-san-economy` pilotes.

REMARQUE	Le `solidfire-san` driver est également capable de cloner des PVC.

Pilotes backend Cloud Volumes ONTAP

Cloud Volumes ONTAP offre le contrôle des données ainsi que des fonctionnalités de stockage de niveau entreprise pour divers cas d'utilisation, notamment le partage de fichiers et le stockage en mode bloc prenant en charge les protocoles NAS et SAN (NFS, SMB / CIFS et iSCSI). Les pilotes compatibles pour Cloud Volume ONTAP sont `ontap-nas`, `ontap-nas-economy`, `ontap-san` et `ontap-san-economy`. Ceux-ci sont applicables pour Cloud Volume ONTAP pour Azure, Cloud Volume ONTAP pour GCP.

Amazon FSx for ONTAP pilotes backend

Amazon FSx for NetApp ONTAP vous permet de tirer parti des fonctionnalités, des performances et des capacités d'administration NetApp que vous connaissez, tout en bénéficiant de la simplicité, de l'agilité, de la sécurité et de l'évolutivité du stockage des données sur AWS. FSx for ONTAP prend en charge de nombreuses fonctionnalités du système de fichiers ONTAP et des API d'administration. Les pilotes compatibles pour Cloud Volume ONTAP sont `ontap-nas`, `ontap-nas-economy`, `ontap-nas-flexgroup`, `ontap-san` et `ontap-san-economy`.

NetApp HCI/SolidFire pilotes backend

Le `solidfire-san` pilote utilisé avec les plateformes NetApp HCI/SolidFire aide l'administrateur à configurer un backend Element pour Trident sur la base des limites de QoS. Si vous souhaitez concevoir votre backend pour définir des limites de QoS spécifiques sur les volumes provisionnés par Trident, utilisez le paramètre `type` dans le fichier backend. L'administrateur peut également restreindre la taille du volume pouvant être créé sur le stockage à l'aide du paramètre `limitVolumeSize`. Actuellement, les fonctionnalités de stockage Element telles que le redimensionnement de volume et la réplication de volume ne sont pas prises en charge via le pilote `solidfire-san`. Ces opérations doivent être effectuées manuellement via l'interface web

d'Element Software.

Pilote SolidFire	Snapshots	Clones	Multi-attach	CHAP	QoS	Redimensionner	Réplication
`solidfire-san`	Oui	Oui	Oui	Oui	Oui	Oui	Oui

> Note de bas de page : Oui [1] : Non géré par Trident Oui [2] : Pris en charge pour les volumes de blocs bruts

Pilotes backend Azure NetApp Files

Trident utilise le `azure-netapp-files` driver pour gérer le "Azure NetApp Files" service.

Vous trouverez plus d'informations sur ce pilote et sur la manière de le configurer dans "Configuration du backend Trident pour Azure NetApp Files".

Pilote Azure NetApp Files	Snapshots	Clones	Multi-attach	QoS	Développer	Réplication
`azure-netapp-files`	Oui	Oui	Oui	Oui	Oui	Oui

> Note de bas de page : Oui [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 explique comment concevoir une classe de stockage pour votre application.

Utilisation spécifique du backend

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. Trois ensembles de filtres peuvent être définis dans la classe de stockage : `storagePools`, `additionalStoragePools`, et/ou `excludeStoragePools`.

Le paramètre `storagePools` permet de restreindre le stockage à l'ensemble des pools correspondant à tout attribut spécifié. Le paramètre `additionalStoragePools` est utilisé pour étendre l'ensemble des pools que Trident utilise pour le provisionnement, ainsi que l'ensemble des pools sélectionnés par les attributs et les paramètres `storagePools`. Vous pouvez utiliser chaque paramètre seul ou les deux ensemble pour vous assurer que l'ensemble approprié de pools de stockage est sélectionné.

Le `excludeStoragePools` paramètre est utilisé pour exclure spécifiquement l'ensemble des pools listés qui correspondent aux attributs.

Émuler les politiques QoS

Si vous souhaitez concevoir des classes de stockage pour émuler des politiques de qualité de service, créez une classe de stockage avec l'attribut `media` comme `hdd` ou `ssd`. En fonction de l'attribut `media` mentionné dans la classe de stockage, Trident sélectionnera le backend approprié qui fournit des agrégats `hdd` ou `ssd` pour correspondre à l'attribut media, puis dirigera le provisionnement des volumes vers l'agrégat spécifique. Nous pouvons donc créer une classe de stockage PREMIUM avec l'attribut `media` défini comme `ssd`, ce qui

pourrait être classé comme la politique QoS PREMIUM. Nous pouvons créer une autre classe de stockage STANDARD avec l'attribut media défini comme `hdd`, ce qui pourrait être classé comme la politique QoS STANDARD. Nous pourrions également utiliser l'attribut ``IOPS'' dans la classe de stockage pour rediriger le provisionnement vers un appareil Element, qui peut être défini comme une politique QoS.

Utiliser le backend en fonction de fonctionnalités spécifiques

Les classes de stockage peuvent être conçues pour diriger l'allocation de volumes sur un backend spécifique où des fonctionnalités telles que l'allocation dynamique et statique, les snapshots, les clones et le chiffrement sont activés. Pour spécifier le stockage à utiliser, créez des classes de stockage qui spécifient le backend approprié avec la fonctionnalité requise activée.

Pools virtuels

Les pools virtuels sont disponibles pour tous les backends Trident. Vous pouvez définir des pools virtuels pour n'importe quel backend, en utilisant n'importe quel pilote que Trident fournit.

Les pools virtuels permettent à un administrateur de créer un niveau d'abstraction au-dessus des backends, référençables via les Storage Classes, pour une plus grande flexibilité et un placement efficace des volumes sur les backends. Différents backends peuvent être définis avec la même classe de service. De plus, plusieurs pools de stockage peuvent être créés sur un même backend mais avec des caractéristiques différentes. Lorsqu'une Storage Class est configurée avec un sélecteur comportant des étiquettes spécifiques, Trident choisit un backend correspondant à toutes les étiquettes du sélecteur pour y placer le volume. Si les étiquettes du sélecteur de la Storage Class correspondent à plusieurs pools de stockage, Trident en choisira un pour provisionner le volume.

Conception de pool virtuel

Lors de la création d'un backend, il est généralement possible de spécifier un ensemble de paramètres. Il était impossible pour l'administrateur de créer un autre backend avec les mêmes informations d'identification de stockage et avec un ensemble de paramètres différent. Avec l'introduction des pools virtuels, ce problème a été atténué. Un pool virtuel est un niveau d'abstraction introduit entre le backend et la classe de stockage Kubernetes afin que l'administrateur puisse définir des paramètres ainsi que des étiquettes qui peuvent être référencées via les classes de stockage Kubernetes comme sélecteur, de manière indépendante du backend. Les pools virtuels peuvent être définis pour tous les backends NetApp pris en charge avec Trident. Cette liste inclut SolidFire/NetApp HCI, ONTAP, ainsi que 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 de backend. Il est également conseillé de ne pas 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. En utilisant l'implémentation de pool virtuel pour Cloud Volume Service for Azure NetApp Files, examinons comment nous pouvons configurer différentes classes de service. Configurez le backend Azure NetApp Files avec plusieurs étiquettes, représentant différents niveaux de performance. Définissez l'aspect `servicelevel` sur le niveau de performance approprié et ajoutez les autres aspects requis sous chaque étiquette. Créez maintenant différentes classes de stockage Kubernetes qui correspondront à différents pools virtuels. À l'aide du champ `parameters.selector`, chaque StorageClass 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 avec un ensemble spécifique d'aspects à partir d'un seul backend de stockage. Pour ce faire, configurez le backend avec plusieurs étiquettes et définissez les aspects requis sous chaque étiquette. Créez ensuite différentes classes de stockage Kubernetes en utilisant le champ `parameters.selector` qui correspondrait à différents pools virtuels. Les volumes provisionnés sur le backend auront les aspects définis dans le pool virtuel choisi.

Caractéristiques du PVC qui affectent le provisionnement du stockage

Certains paramètres autres que la classe de stockage demandée peuvent affecter le processus de décision de provisionnement Trident lors de la création d'un PVC.

Mode d'accès

Lors d'une demande de stockage via un PVC, le mode d'accès est un champ obligatoire. Le mode souhaité peut affecter le backend sélectionné pour héberger la demande de stockage.

Trident tentera d'associer le protocole de stockage utilisé à la méthode d'accès spécifiée selon la matrice suivante. Ceci est indépendant de la plateforme de stockage sous-jacente.

	ReadWriteOnce	ReadOnlyMany	ReadWriteMany
iSCSI	Oui	Oui	Oui (Raw block)
NFS	Oui	Oui	Oui

Une demande de PVC ReadWriteMany soumise à un déploiement Trident sans backend NFS configuré n'entraînera pas la création de volume. Pour cette raison, le demandeur doit utiliser le mode d'accès approprié pour son application.

Opérations de volume

Modifier les volumes persistants

Dans Kubernetes, les volumes persistants sont, à deux exceptions près, des objets immuables. Une fois créés, la politique de récupération et la taille peuvent être modifiées. Cependant, cela n'empêche pas que certains aspects du volume soient modifiés en dehors de Kubernetes. Cela peut être souhaitable afin de personnaliser le volume pour des applications spécifiques, de s'assurer que la capacité n'est pas accidentellement consommée, ou simplement de déplacer le volume vers un autre contrôleur de stockage pour n'importe quelle raison.

REMARQUE	Les provisionneurs intégrés de Kubernetes ne prennent pas en charge les opérations de redimensionnement de volumes pour les volumes persistants NFS, iSCSI ou FC 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 création.

Créer des instantanés de volume à la demande

Trident prend en charge la création à la demande d'instantanés de volumes et la création de PVC à partir d'instantanés grâce au framework CSI. Les instantanés constituent une méthode pratique pour conserver des copies des données à un instant précis et ont un cycle de vie indépendant du PV source dans Kubernetes. Ces instantanés peuvent être utilisés pour cloner des PVC.

Créer des volumes à partir d'instantanés

Trident prend également en charge la création de PersistentVolumes à partir d'instantanés de volume. Pour ce faire, il suffit de créer un PersistentVolumeClaim et de mentionner l' `datasource`instantané requis à partir duquel le volume doit être créé. Trident gérera ce PVC en créant un volume avec les données présentes sur l'instantané. Grâce à cette fonctionnalité, il est possible de dupliquer des données entre régions, de créer des environnements de test, de remplacer intégralement un volume de production endommagé ou corrompu, ou de récupérer des fichiers et répertoires spécifiques et de les transférer vers un autre volume attaché.

Déplacer les volumes dans le cluster

Les administrateurs de stockage ont la possibilité de déplacer des volumes entre les agrégats et les contrôleurs dans le cluster ONTAP sans interruption pour le consommateur de stockage. Cette opération n'affecte pas Trident ni le cluster Kubernetes, tant que l'agrégat de destination est accessible par la SVM que Trident utilise. Il est important de noter que si l'agrégat a été nouvellement ajouté à la SVM, le backend devra être actualisé en le réajoutant à Trident. Cela déclenchera une réinvention de la SVM par Trident afin que le nouvel agrégat soit reconnu.

Cependant, le déplacement de volumes entre systèmes de stockage n'est pas pris en charge automatiquement par Trident. Cela inclut les SVM au sein d'un même cluster, entre clusters, ou vers 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 volumes peut être utilisée pour importer les volumes actuels dans Trident.

Étendre les volumes

Trident prend en charge le redimensionnement des PV NFS, iSCSI et FC. Cela permet aux utilisateurs de redimensionner leurs volumes directement via la couche Kubernetes. L'expansion de volume est possible pour toutes les principales plateformes de stockage NetApp, y compris ONTAP, et les backends SolidFire/NetApp HCI. Pour permettre une expansion ultérieure, définissez `allowVolumeExpansion` sur `true` dans votre StorageClass associée au volume. Chaque fois que le volume persistant doit être redimensionné, modifiez l'annotation `spec.resources.requests.storage` dans la revendication de volume persistant à la taille de volume requise. Trident prendra automatiquement en charge le redimensionnement du volume sur le cluster de stockage.

Importer un volume existant dans Kubernetes

L'importation de volumes permet d'importer un volume de stockage existant dans un environnement Kubernetes. Cette fonctionnalité est actuellement prise en charge par les `ontap-nas`, `ontap-nas-flexgroup`, `solidfire-san` et `azure-netapp-files` pilotes. Cette fonctionnalité est utile lors de la migration d'une application existante vers Kubernetes ou dans des scénarios de reprise après sinistre.

Lors de l'utilisation des pilotes ONTAP et `solidfire-san`, utilisez la commande `tridentctl import volume <backend-name> <volume-name> -f /path/pvc.yaml` pour importer un volume existant dans Kubernetes afin qu'il soit géré par Trident. Le fichier PVC YAML ou JSON utilisé dans la commande d'importation de volume pointe vers une classe de stockage qui identifie Trident comme le provisionneur. Lorsque vous utilisez un backend NetApp HCI/SolidFire, assurez-vous que les noms de volumes sont uniques. Si les noms de volumes sont dupliqués, clonez le volume sous un nom unique afin que la fonctionnalité d'importation de volumes puisse les distinguer.

Si le `azure-netapp-files` driver 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.

Lorsque la commande ci-dessus est exécutée, Trident détecte le volume sur le backend et lit sa taille. Il ajoute automatiquement (et écrase si nécessaire) la taille du volume du PVC configuré. Trident crée ensuite le nouveau PV et Kubernetes lie le PVC au PV.

Si un conteneur a été déployé de manière à nécessiter le PVC importé spécifique, il restera en attente jusqu'à ce que la paire PVC/PV soit liée via le processus d'importation de volume. Après la liaison de la paire PVC/PV, le conteneur devrait démarrer, 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 OpenShift services, le service de journalisation est déployé à l'aide d'Ansible avec des paramètres de configuration fournis par le fichier d'inventaire, également appelé hosts, fourni au playbook. Il existe deux méthodes d'installation qui seront abordées : le déploiement de la journalisation lors de l'installation initiale de OpenShift et le déploiement de la journalisation après que OpenShift a été installé.

AVERTISSEMENT	À partir de la version 3.9 de Red Hat OpenShift, la documentation officielle déconseille l'utilisation de NFS pour le service de journalisation en raison de préoccupations concernant la corruption des données. Cette recommandation repose sur les tests effectués par Red Hat sur ses produits. Le serveur NFS ONTAP ne présente pas ces problèmes et peut facilement prendre en charge un déploiement de journalisation. En définitive, le choix du protocole pour le service de journalisation vous appartient, sachez simplement que les deux fonctionneront parfaitement avec les plateformes NetApp et qu'il n'y a aucune raison d'éviter NFS si c'est votre préférence.

Si vous choisissez d'utiliser NFS avec le service de journalisation, vous devrez définir la variable Ansible `openshift_enable_unsupported_configurations` sur `true` pour empêcher l'installateur d'échouer.

Commencer

Le service de journalisation peut, en option, être déployé à la fois pour les applications ainsi que pour les opérations principales du OpenShift cluster lui-même. 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 « ops », tandis que l'instance pour les applications ne les contient pas.

Il est important de configurer les variables Ansible en fonction de la méthode de déploiement afin de garantir que le stockage approprié est utilisé par les services sous-jacents. Examinons les options pour chacune des méthodes de déploiement.

REMARQUE	Les tableaux ci-dessous ne contiennent que les variables pertinentes pour la configuration du stockage en lien avec le service de journalisation. Vous pouvez trouver d'autres options dans "Documentation de journalisation Red Hat OpenShift" qui doivent être examinées, configurées et utilisées en fonction de votre déploiement.

Les variables du tableau ci-dessous permettront au playbook Ansible de créer un PV et un PVC pour le service de journalisation en utilisant les informations fournies. Cette méthode est nettement moins flexible que l'utilisation du playbook d'installation du composant après l'installation de OpenShift, cependant, si vous disposez de volumes existants, c'est une option.

Variable	Détails
`openshift_logging_storage_kind`	Définissez 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. Cela doit être défini sur la dataLIF de votre machine virtuelle.
`openshift_logging_storage_nfs_directory`	Chemin de montage pour l'export NFS. Par exemple, si le volume est monté comme `/openshift_logging`, vous utiliserez ce chemin pour cette variable.
`openshift_logging_storage_volume_name`	Le nom, par exemple `pv_ose_logs`, du PV à créer.
`openshift_logging_storage_volume_size`	La taille de l'export NFS, par exemple `100Gi`.

Si votre OpenShift cluster est déjà en cours d'exécution, et que Trident est donc déployé et configuré, le programme d'installation peut utiliser le provisionnement dynamique pour créer les volumes. Les variables suivantes devront être configurées.

Variable	Détails
`openshift_logging_es_pvc_dynamic`	Définissez sur true pour utiliser des volumes provisionnés dynamiquement.
`openshift_logging_es_pvc_storage_class_name`	Le nom de la classe de stockage qui sera utilisé dans le PVC.
`openshift_logging_es_pvc_size`	La taille du volume demandée dans le PVC.
`openshift_logging_es_pvc_prefix`	Un préfixe pour les PVC utilisés par le service de journalisation.
`openshift_logging_es_ops_pvc_dynamic`	Définissez sur `true` pour utiliser des volumes provisionnés dynamiquement pour l'instance de journalisation des opérations.
`openshift_logging_es_ops_pvc_storage_class_name`	Le nom de la classe de stockage pour l'instance de journalisation des opérations.
`openshift_logging_es_ops_pvc_size`	La taille de la requête de volume pour l'instance ops.
`openshift_logging_es_ops_pvc_prefix`	Un préfixe pour les PVC d'instance ops.

Déployez la pile de journalisation

Si vous déployez la journalisation dans le cadre du processus d'installation initiale de OpenShift, il vous suffit de suivre la procédure de déploiement standard. Ansible configurera et déploiera les services et objets OpenShift nécessaires pour que le service soit disponible dès qu'Ansible a terminé.

Toutefois, si vous effectuez un déploiement après l'installation initiale, le playbook du composant devra être utilisé par Ansible. Ce processus peut légèrement varier selon les différentes versions de OpenShift, alors assurez-vous de lire et de suivre "Documentation de Red Hat OpenShift Container Platform 3.11" pour votre version.

Service de métriques

Le service de métriques fournit à l'administrateur des informations précieuses concernant l'état, l'utilisation des ressources et la disponibilité du OpenShift cluster. Il est également nécessaire pour la fonctionnalité d'auto-scaling des pods et de nombreuses organisations utilisent les données du service de métriques pour leurs applications de refacturation et/ou de transparence des coûts.

Comme pour le service de journalisation, et OpenShift dans son ensemble, Ansible est utilisé pour déployer le service de métriques. De même que pour le service de journalisation, le service de métriques peut être déployé lors de la configuration initiale du cluster ou après sa mise en service en utilisant la méthode d'installation des composants. Les tableaux suivants contiennent les variables qui sont importantes lors de la configuration du stockage persistant pour le service de métriques.

REMARQUE	Les tableaux ci-dessous ne contiennent que les variables pertinentes pour la configuration du stockage en lien avec le service de métriques. De nombreuses autres options sont décrites dans la documentation, qui doivent être examinées, configurées et utilisées selon votre déploiement.

Variable	Détails
`openshift_metrics_storage_kind`	Définissez 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. Cela doit être défini sur la dataLIF de votre SVM.
`openshift_metrics_storage_nfs_directory`	Chemin de montage pour l'export NFS. Par exemple, si le volume est monté comme `/openshift_metrics`, vous utiliserez ce chemin pour cette variable.
`openshift_metrics_storage_volume_name`	Le nom, par exemple `pv_ose_metrics`, du PV à créer.
`openshift_metrics_storage_volume_size`	La taille de l'export NFS, par exemple `100Gi`.

Si votre OpenShift cluster est déjà en cours d'exécution, et que Trident est donc déployé et configuré, le programme d'installation peut utiliser le provisionnement dynamique pour créer les volumes. Les variables suivantes devront être configurées.

Variable	Détails
`openshift_metrics_cassandra_pvc_prefix`	Un préfixe à utiliser pour les PVC de métriques.
`openshift_metrics_cassandra_pvc_size`	La taille des volumes à demander.
`openshift_metrics_cassandra_storage_type`	Le type de stockage à utiliser pour les métriques, cela doit être défini sur dynamique pour qu'Ansible puisse créer des PVC avec la classe de stockage appropriée.
`openshift_metrics_cassanda_pvc_storage_class_name`	Le nom de la classe de stockage à utiliser.

Déployez le service de métriques

Une fois les variables Ansible appropriées définies dans votre fichier hosts/inventory, déployez le service à l'aide d'Ansible. Si vous effectuez le déploiement au moment de l'installation de OpenShift, alors le PV sera créé et utilisé automatiquement. Si vous déployez à l'aide des playbooks de composants, après l'installation de OpenShift, alors Ansible crée les PVC nécessaires et, après que Trident a provisionné le stockage pour eux, déploie le service.

Les variables ci-dessus, ainsi que le processus de déploiement, peuvent changer avec chaque version de OpenShift. Assurez-vous de consulter et de suivre "Guide de déploiement OpenShift de Red Hat" pour votre version afin qu'elle soit configurée pour votre environnement.

Protection des données et reprise après sinistre

Découvrez les options de protection et de récupération pour Trident et les volumes créés avec Trident. Vous devez disposer d'une stratégie de protection des données et de récupération pour chaque application nécessitant une persistance.

Réplication et récupération de Trident

Vous pouvez créer une sauvegarde pour restaurer Trident en cas de sinistre.

Réplication Trident

Trident utilise les CRD Kubernetes pour stocker et gérer son propre état et le cluster Kubernetes etcd pour stocker ses métadonnées.

Étapes

1. Sauvegardez le cluster Kubernetes etcd en utilisant "Kubernetes : sauvegarde d'un cluster etcd".
2. Placez les fichiers de sauvegarde sur un volume FlexVol

REMARQUE	NetApp recommande de protéger la SVM où le volume FlexVol réside avec une relation SnapMirror vers une autre SVM.

Récupération de Trident

En utilisant les CRD Kubernetes et l'instantané etcd du cluster Kubernetes, vous pouvez récupérer Trident.

Étapes

1. Depuis la SVM de destination, montez le volume qui contient les fichiers de données etcd de Kubernetes et les certificats sur l'hôte qui sera configuré comme nœud maître.
2. Copiez tous les certificats requis relatifs au 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 en utilisant "Kubernetes : restauration d'un cluster etcd".
4. Exécutez `kubectl get crd` pour vérifier que toutes les ressources personnalisées Trident sont opérationnelles et récupérez les objets Trident pour vérifier que toutes les données sont disponibles.

Réplication et récupération de SVM

Trident ne peut pas configurer les relations de réplication, cependant, l'administrateur de stockage peut utiliser "ONTAP SnapMirror" pour répliquer une SVM.

En cas de sinistre, vous pouvez activer le SVM de destination SnapMirror pour commencer à servir les données. Vous pouvez revenir au SVM principal une fois les systèmes rétablis.

À propos de cette tâche

Tenez compte des points suivants lors de l'utilisation de la fonctionnalité de réplication SVM SnapMirror :

- Vous devez créer un backend distinct pour chaque SVM avec SVM-DR activé.
- Configurez les classes de stockage pour sélectionner les backends répliqués uniquement lorsque cela est nécessaire afin d'éviter que des volumes qui n'ont pas besoin de réplication soient provisionnés sur les backends qui prennent en charge SVM-DR.
- Les administrateurs d'applications doivent comprendre les coûts et la complexité supplémentaires associés à la réplication et examiner attentivement leur plan de reprise avant de commencer ce processus.

Réplication SVM

Vous pouvez utiliser "ONTAP : SnapMirror réplication SVM" pour créer la relation de réplication SVM.

SnapMirror vous permet de définir des options pour contrôler ce qui doit être répliqué. Vous devrez connaître les options que vous avez sélectionnées lors de l'exécution Récupération SVM à l'aide de Trident.

- "-identity-preserve true"réplique l'intégralité de la configuration SVM.
- "-discard-configs réseau" exclut les LIF et les paramètres réseau associés.
- "-identity-preserve false" Réplique uniquement les volumes et la configuration de sécurité.

Récupération SVM à l'aide de Trident

Trident ne détecte pas automatiquement les pannes de SVM. En cas de sinistre, l'administrateur peut déclencher manuellement le basculement de Trident vers le nouveau SVM.

Étapes

1. Annulez les transferts SnapMirror planifiés et en cours, interrompez la relation de réplication, arrêtez la SVM source puis activez la SVM de destination SnapMirror.
2. Si vous avez spécifié `-identity-preserve false` ou `-discard-config network` lors de la configuration de votre réplication SVM, mettez à jour `managementLIF` et `dataLIF` dans le fichier de définition du backend Trident.
3. Confirmer `storagePrefix` est présent dans le fichier de définition du backend Trident. Ce paramètre ne peut pas être modifié. Omettre `storagePrefix` entraînera l'échec de la mise à jour du backend.
4. Mettez à jour tous les backends requis pour refléter le nouveau nom de la SVM de destination en utilisant :

```
./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 redémarrer tous les pods d'application.

REMARQUE	Si vous avez spécifié `-identity-preserve true`, tous les volumes provisionnés par Trident commencent à servir des données lorsque le SVM de destination est activé.

Réplication et récupération de volume

Trident ne peut pas configurer les relations de réplication SnapMirror, cependant, l'administrateur de stockage peut utiliser "Réplication et récupération ONTAP SnapMirror" pour répliquer les volumes créés par Trident.

Vous pouvez ensuite importer les volumes récupérés dans Trident en utilisant "tridentctl volume import".

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 des données à l'aide de :

- Un contrôleur de snapshots externe et des CRD pour créer des snapshots de volumes persistants (PVs) Kubernetes.

  "Instantanés de volume"

- Utilisez les snapshots ONTAP pour restaurer l'intégralité du contenu d'un volume ou pour récupérer des fichiers ou des LUN individuels.

  "Instantanés ONTAP"

Automatisation du basculement des applications avec état avec Trident

La fonctionnalité de détachement forcé de Trident 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 des nœuds deviennent indisponibles ou sont mis hors ligne pour maintenance.

Détails concernant le détachement forcé

Le détachement forcé est disponible pour `ontap-san`, `ontap-san-economy`, `ontap-nas` et `ontap-nas-economy` uniquement. Avant d'activer le détachement forcé, l'arrêt brutal des nœuds (NGNS) doit être activé sur le cluster Kubernetes. NGNS est activé par défaut pour Kubernetes 1.28 et versions ultérieures. Pour plus d'informations, consultez "Kubernetes : arrêt non gracieux d'un nœud".

REMARQUE	Lorsque vous utilisez le `ontap-nas` ou `ontap-nas-economy` pilote, vous devez définir le paramètre `autoExportPolicy` dans la configuration du backend sur `true` afin que Trident puisse restreindre l'accès depuis le nœud Kubernetes avec la contamination appliquée à l'aide de politiques d'exportation gérées.

AVERTISSEMENT	Étant donné que Trident repose sur Kubernetes NGNS, ne supprimez pas `out-of-service` les taints d'un nœud défaillant tant que toutes les charges de travail non tolérables n'ont pas été reprogrammées. Appliquer ou supprimer le taint sans précaution peut compromettre la protection des données du backend.

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

1. Arrêtez l'accès E/S du backend pour les volumes montés sur ce nœud.
2. Marquez l'objet nœud Trident comme `dirty` (non sûr pour les nouvelles publications).

REMARQUE	Le contrôleur Trident refusera les nouvelles demandes de publication de volumes tant que le nœud n'aura pas été requalifié (après avoir été marqué comme `dirty`) par le pod Trident du nœud. Les charges de travail planifiées avec un PVC monté (même après que le nœud de cluster soit opérationnel et prêt) ne seront pas acceptées tant que Trident ne peut pas vérifier le nœud `clean` (sécurisé pour de nouvelles publications).

Lorsque l'intégrité du nœud est rétablie et que la contamination est supprimée, Trident :

1. Identifiez et nettoyez les chemins publiés obsolètes sur le nœud.
2. Si le nœud est dans un `cleanable` état (la contamination hors service a été supprimée et le nœud est dans un `Ready` état) et que tous les chemins publiés obsolètes sont propres, Trident réadmettra le nœud en `clean` et autorisera de nouveaux volumes publiés sur le nœud.

Détails sur le basculement automatique

Vous pouvez automatiser le processus de déconnexion forcée grâce à l'intégration avec "opérateur de vérification de l'état des nœuds (NHC)". Lorsqu'une défaillance de nœud se produit, NHC déclenche la remédiation du nœud Trident (TNR) et la déconnexion forcée automatiquement en créant un CR TridentNodeRemediation dans l'espace de noms de Trident, définissant le nœud défaillant. La TNR est créée uniquement lors d'une défaillance de nœud et supprimée par NHC une fois que le nœud est remis en ligne ou 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 sale, empêchant toute nouvelle publication de volume et commence à supprimer les pods compatibles avec le détachement forcé ainsi que leurs attachements de volumes.

Tous les volumes/PVC pris en charge par force-detach sont pris en charge par automated-failover :

- Volumes NAS et NAS-economy utilisant des politiques d'auto-exportation (SMB n'est pas encore pris en charge).
- Volumes SAN et SAN-economy.

Consultez Détails concernant le détachement forcé.

Comportement par défaut:

- Les pods utilisant des volumes compatibles avec le détachement forcé sont supprimés du nœud défaillant. Kubernetes les reprogrammera sur un nœud sain.
- Les pods utilisant un volume non pris en charge par le force-detach, 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 `trident.netapp.io/podRemediationPolicy: delete` est définie.

Remplacement du comportement de suppression des pods :

Le comportement de suppression des pods peut être personnalisé à l'aide d'une annotation : `trident.netapp.io/podRemediationPolicy[retain, delete]`. Ces annotations sont examinées et utilisées lors d'un basculement. Appliquez des annotations à la spécification du pod du déploiement/replicaset Kubernetes pour éviter que l'annotation ne disparaisse 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 force-detach. • Pour les volumes qui ne prennent pas en charge le détachement forcé, il existe un risque de corruption des données et de problèmes de multi-connexion.

CR TridentNodeRemediation

Le TridentNodeRemediation (TNR) CR définit un nœud défaillant. Le nom du TNR est le nom du nœud défaillant.

Exemple de TNR:

```
apiVersion: trident.netapp.io/v1
kind: TridentNodeRemediation
metadata:
  name: <K8s-node-name>
spec: {}
```

États TNR : Utilisez les commandes suivantes pour afficher l'état des TNR :
`kubectl get tnr <name> -n <trident-namespace>`

Les 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 force-detach montés sur ce nœud.
  - L'objet nœud Trident est marqué comme sale (non sûr pour les nouvelles publications).
  - Supprimez les pods et les attachements de volumes du nœud

- *Récupération du nœud en attente*:
  - Le contrôleur attend que le nœud soit de nouveau en ligne.
  - Une fois le nœud en ligne, publish-enforcement garantira que le nœud est propre et prêt pour de 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 du nœud ont été réalisées avec succès. Le nœud est propre et prêt pour la publication de nouveaux volumes.
- *Échec* :
  - Erreur irrécupérable. Les raisons de l'erreur sont indiquées 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, consultez Détails concernant le détachement forcé.
- Installez la vérification de l'état (NHC) dans le cluster Kubernetes.
  - "Installer operator-sdk".
  - 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 ressource personnalisée 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 le CR de vérification de l'état du nœud dans l'espace de noms `trident`.

```
kubectl apply -f <nhc-cr-file>.yaml -n <trident-namespace>
```

La ressource personnalisée (CR) ci-dessus est configurée pour surveiller les nœuds de travail K8s et détecter les états Ready: false et Unknown. Automated-Failover sera déclenché lorsqu'un nœud passe à l'état Ready: false ou Ready: Unknown.

Le `unhealthyConditions` dans le CR utilise un délai de grâce de 0 seconde. Cela provoque un basculement automatique immédiat dès que K8s définit la condition Ready : false, ce qui se produit après que K8s a perdu le signal de présence d'un nœud. K8s attend par défaut 40 secondes après le dernier signal de présence avant de définir Ready : false. Ce délai de grâce peut être personnalisé dans les options de déploiement de K8s.

Pour plus d'options de configuration, veuillez vous référer à "Documentation de l'opérateur Node-Healthcheck-Operator".

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.

TridentNodeRemediationTemplate (TNRT) :

Le TNRT sert de modèle au contrôleur NHC, qui utilise 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: {}
```

ClusterRole :

Un rôle de cluster est également ajouté lors de l'installation lorsque le détachement forcé est activé. Cela donne à NHC les autorisations 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 la maintenance ou la mise à niveau de K8s, lorsque les nœuds doivent être arrêtés ou redémarrés. Vous pouvez suspendre le CR NHC (décrit ci-dessus) en modifiant son CR :

```
kubectl patch NodeHealthCheck <cr-name> --patch
'{"spec":{"pauseRequests":["<description-for-reason-of-pause>"]}}' --type=merge
```

Cela suspend le basculement automatique. Pour réactiver le basculement automatique, supprimez le pauseRequests de la spécification après la fin de la maintenance.

Limitations

- Les opérations d'E/S sont bloquées uniquement sur les nœuds défaillants pour les volumes pris en charge par force-detach. Seuls les pods utilisant des volumes/PVCs pris en charge par force-detach sont automatiquement supprimés.
- Le basculement automatique et le détachement forcé s'exécutent au sein du pod trident-controller. Si le nœud hébergeant trident-controller tombe en panne, le basculement automatique sera différé jusqu'à ce que K8s déplace le pod vers un nœud sain.

Intégration de solutions personnalisées de vérification de l'état des nœuds

Vous pouvez remplacer Node Healthcheck Operator 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 du CR TNR.
- Supprimez le TNR lorsque le nœud a récupéré et que le TNR est à l'état Succeeded.

Sécurité

Sécurité

Utilisez les recommandations listées ici pour garantir 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 de garantir un stockage fiable et de bloquer toute activité malveillante potentielle.

Pour séparer les autres applications et utilisateurs de Trident, installez toujours Trident dans son propre espace de noms Kubernetes (`trident`). Mettre Trident dans son propre espace de noms garantit que seul le personnel administratif de Kubernetes a accès au pod Trident et aux artefacts (tels que les secrets backend et CHAP, le cas échéant) stockés dans les objets CRD de l'espace de noms. Vous devez vous assurer que seuls les administrateurs ont accès à l'espace de noms Trident et donc à l'application `tridentctl`.

Utilisez l'authentification CHAP avec les backends SAN ONTAP

Trident prend en charge l'authentification CHAP pour les charges de travail ONTAP SAN (à l'aide des pilotes `ontap-san` et `ontap-san-economy`). NetApp recommande d'utiliser l'authentification CHAP bidirectionnelle avec Trident pour l'authentification entre un hôte et le système de stockage.

Pour les backends ONTAP qui utilisent les pilotes de stockage SAN, Trident peut configurer le CHAP bidirectionnel et gérer les noms d'utilisateur et les secrets CHAP via `tridentctl`. Reportez-vous à

"Préparez-vous à configurer le backend avec les pilotes SAN ONTAP" pour comprendre comment Trident configure CHAP sur les backends ONTAP.

Utilisez l'authentification CHAP avec NetApp HCI et SolidFire backends

NetApp recommande de déployer CHAP bidirectionnel pour garantir l'authentification entre un hôte et les backends 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 objet CR `tridentvolume` pour le PV respectif. Lorsque vous créez un PV, Trident utilise les secrets CHAP pour initier 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 Volume Access Group.

Utilisez Trident avec NVE et NAE

NetApp ONTAP assure le chiffrement des données au repos afin de protéger les données sensibles en cas de vol, de retour ou de réutilisation d'un disque. Pour plus de détails, consultez "Configurer l'aperçu du chiffrement des volumes NetApp".

- Si NAE est activé sur le backend, tout volume provisionné dans Trident sera activé NAE.
  - Vous pouvez définir le indicateur de chiffrement NVE sur `""` pour créer des volumes compatibles NAE.
- Si NAE n'est pas activé sur le backend, tout volume provisionné dans Trident sera activé NVE à moins que l'indicateur de chiffrement NVE ne soit défini sur `false` (la valeur par défaut) dans la configuration du backend.

REMARQUE	Les volumes créés dans Trident sur un backend activé NAE doivent être chiffrés NVE ou NAE. • Vous pouvez définir le indicateur de chiffrement NVE sur `true` dans la configuration du backend Trident pour remplacer le chiffrement NAE et utiliser une clé de chiffrement spécifique pour chaque volume. • Définir le drapeau de chiffrement NVE sur `false` sur un backend activé NAE crée un volume activé NAE. Vous ne pouvez pas désactiver le chiffrement NAE en définissant le drapeau de chiffrement NVE sur `false`.

- Vous pouvez créer manuellement un volume NVE dans Trident en définissant explicitement le indicateur de chiffrement NVE sur `true`.

Pour plus d'informations sur les options de configuration du backend, consultez :

- "Options de configuration SAN ONTAP"
- "Options de configuration NAS ONTAP"

Linux Unified Key Setup (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 des phrases de passe et l'extension de volume pour les volumes chiffrés avec LUKS.

Dans Trident, les volumes chiffrés LUKS utilisent le cypher et le mode aes-xts-plain64, comme recommandé

par "NIST".

REMARQUE	Le chiffrement LUKS n'est pas pris en charge pour les systèmes ASA r2. Pour des informations sur les systèmes ASA r2, voir "En savoir plus sur les systèmes de stockage ASA r2".

Avant de commencer

- Les nœuds de travail doivent avoir cryptsetup 2.1 ou une version supérieure (mais inférieure à 3.0) installée. Pour plus d'informations, consultez "Gitlab : cryptsetup".
- Pour des raisons de performance, NetApp recommande que les nœuds de travail prennent en charge Advanced Encryption Standard New Instructions (AES-NI). Pour vérifier la prise en charge d'AES-NI, exécutez la commande suivante :

  ```
  grep "aes" /proc/cpuinfo
  ```

  Si aucune réponse n'est renvoyée, votre processeur ne prend pas en charge AES-NI. Pour plus d'informations sur AES-NI, visitez : "Intel: Instructions Advanced Encryption Standard (AES-NI)".

Activer le chiffrement LUKS

Vous pouvez activer le chiffrement par volume, côté hôte, à l'aide de Linux Unified Key Setup (LUKS) pour les volumes ONTAP SAN et ONTAP SAN ECONOMY.

Étapes

1. Définissez les attributs de chiffrement LUKS dans la configuration du backend. Pour plus d'informations sur les options de configuration du backend pour ONTAP SAN, consultez "Options de configuration SAN ONTAP".

```
{
  "storage": [
    {
      "labels": {
        "luks": "true"
      },
      "zone": "us_east_1a",
      "defaults": {
        "luksEncryption": "true"
      }
    },
    {
      "labels": {
        "luks": "false"
      },
      "zone": "us_east_1a",
      "defaults": {
        "luksEncryption": "false"
      }
    }
  ]
}
```

2. Utilisez `parameters.selector` pour définir les pools de stockage utilisant le chiffrement 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
```

Limitations

Les volumes chiffrés LUKS ne peuvent pas bénéficier de la déduplication et de la compression ONTAP.

Configuration du backend pour l'importation des volumes LUKS

Pour importer un volume LUKS, vous devez définir `luksEncryption` sur (`true` dans le backend. L' `luksEncryption`option indique à Trident si le volume est conforme LUKS (`true`) ou non conforme LUKS (`false`), comme illustré 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 dynamiquement des volumes LUKS, définissez l'annotation `trident.netapp.io/luksEncryption` sur `true` et incluez une classe de stockage compatible LUKS dans le PVC 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
```

Faire pivoter une phrase de passe LUKS

Vous pouvez faire pivoter la phrase de passe LUKS et confirmer la rotation.

AVERTISSEMENT	N'oubliez pas votre phrase de passe tant que vous n'avez pas vérifié qu'elle n'est plus utilisée par aucun volume, instantané ou secret. Si une phrase de passe référencée est perdue, vous pourriez être dans l'incapacité de monter le volume et les données resteront chiffrées et inaccessibles.

À propos de cette tâche

La rotation de la phrase de passe LUKS a lieu 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 du volume à la phrase de passe active dans le secret.

- Si la phrase de passe sur le volume ne correspond pas à la phrase de passe active dans le secret, une rotation a lieu.
- Si la phrase de passe du volume correspond à la phrase de passe active du secret, le `previous-luks-passphrase` paramètre est ignoré.

Étapes

1. Ajoutez les `node-publish-secret-name` et `node-publish-secret-namespace` paramètres StorageClass. 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. Identifiez les phrases de passe existantes sur le volume ou le snapshot.

Volume

```
tridentctl -d get volume luks-pvc1
GET http://127.0.0.1:8000/trident/v1/volume/<volumeID>

...luksPassphraseNames:["A"]
```

Instantané

```
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 du volume en spécifiant la nouvelle et l'ancienne phrase de passe. Assurez-vous que `previous-luke-passphrase-name` et `previous-luks-passphrase` correspondent à 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 en montant le volume. Ceci est nécessaire pour initier la rotation.
5. Vérifiez que la phrase de passe a été modifiée.

Volume

```
tridentctl -d get volume luks-pvc1
GET http://127.0.0.1:8000/trident/v1/volume/<volumeID>

...luksPassphraseNames:["B"]
```

Instantané

```
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 renouvelée lorsque seule la nouvelle phrase de passe est renvoyée sur le volume et l'instantané.

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'expansion du volume

Vous pouvez activer l'extension de volume sur un volume chiffré LUKS.

Étapes

1. Activez la `CSINodeExpandSecret` feature gate (bêta 1.25+). Consultez "Kubernetes 1.25 : Utilisez des secrets pour l'extension des volumes CSI pilotée par les nœuds" pour plus de détails.
2. Ajoutez les `node-expand-secret-name` et `node-expand-secret-namespace` paramètres StorageClass. 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 lancez une extension de stockage en ligne, le kubelet transmet les informations d'identification appropriées au driver.

Chiffrement Kerberos en vol

En utilisant le chiffrement Kerberos in-flight, vous pouvez améliorer la sécurité d'accès aux données en activant le chiffrement pour le trafic entre votre cluster géré et le stockage backend.

Trident prend en charge le chiffrement Kerberos pour ONTAP en tant que backend 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, prendre un instantané, cloner, cloner en lecture seule et importer des volumes qui utilisent le chiffrement NFS.

Configurer le chiffrement Kerberos en transit avec les volumes ONTAP sur site

Vous pouvez activer le chiffrement Kerberos sur le trafic de stockage entre votre cluster géré et un système de stockage ONTAP sur site.

REMARQUE	Le chiffrement Kerberos pour le trafic NFS avec des backends de stockage ONTAP sur site n'est pris en charge qu'à l'aide du `ontap-nas` storage driver.

Avant de commencer

- Assurez-vous d'avoir accès à l' `tridentctl`utilitaire.
- Assurez-vous de disposer d'un accès administrateur au système de stockage ONTAP.
- Assurez-vous de connaître le nom du ou des volumes que vous partagerez depuis le stockage backend ONTAP.
- Assurez-vous d'avoir préparé la machine virtuelle de stockage ONTAP pour prendre en charge le chiffrement Kerberos pour les volumes NFS. Consultez "Activer Kerberos sur une dataLIF" 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".

Ajouter ou modifier les règles d'export ONTAP

Vous devez ajouter des règles aux règles d'export ONTAP existantes ou créer de nouvelles règles d'export prenant en charge le chiffrement Kerberos pour le volume racine de la machine virtuelle de stockage ONTAP, ainsi que pour tous les volumes ONTAP partagés avec le cluster Kubernetes en amont. Les règles d'export que vous ajoutez, ou les nouvelles règles d'export que vous créez, doivent prendre en charge les protocoles d'accès et les autorisations d'accès suivants :

Protocoles d'accès

Configurez la règle d'export avec les protocoles d'accès NFS, NFSv3 et NFSv4.

Détails d'accès

Vous pouvez configurer l'une des trois versions différentes du chiffrement Kerberos, en fonction de vos

besoins pour le volume :

- **Kerberos 5** - (authentification et chiffrement)
- **Kerberos 5i** - (authentification et chiffrement avec protection de l'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 doivent monter les volumes NFS avec un mélange de chiffrement 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

Consultez la documentation suivante pour savoir comment créer des règles d'export ONTAP et des règles de règles d'export :

- "Créer une règles d'export"
- "Ajouter une règle à une règle d'export"

Créer un backend de stockage

Vous pouvez créer une configuration de stockage Trident qui inclut la capacité de chiffrement Kerberos.

À propos de cette tâche

Lorsque vous créez un fichier de configuration du 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 vie privée)

Spécifiez un seul niveau Kerberos. Si vous spécifiez plusieurs niveaux de chiffrement Kerberos dans la liste des paramètres, seule la première option est utilisée.

Étapes

1. Sur le cluster géré, créez un fichier de configuration du backend de stockage en utilisant l'exemple suivant. Remplacez les valeurs entre crochets <> avec 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 backend échoue, cela signifie qu'il y a un problème avec la configuration du backend. Vous pouvez consulter les journaux pour en 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 à nouveau la commande create.

Créer une classe de stockage

Vous pouvez créer une classe de stockage pour provisionner des volumes avec chiffrement Kerberos.

À propos de cette tâche

Lorsque vous créez un objet de 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 vie privée)

Spécifiez un seul niveau Kerberos. Si vous spécifiez plusieurs niveaux de chiffrement 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 backend de stockage est différent de celui que vous spécifiez dans l'objet de classe de stockage, l'objet de classe de stockage prévaut.

Étapes

1. Créez un objet Kubernetes StorageClass en utilisant 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. Créez 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 devriez voir une sortie similaire à la suivante :

```
NAME          PROVISIONER             AGE
ontap-nas-sc    csi.trident.netapp.io   15h
```

Approvisionner des volumes

Après avoir créé un système de stockage et une classe de stockage, vous pouvez maintenant provisionner un volume. Pour obtenir des instructions, consultez "Provisionner un volume".

Configurer le chiffrement Kerberos en transit avec les volumes Azure NetApp Files

Vous pouvez activer le chiffrement Kerberos sur le trafic de stockage entre votre cluster géré et un seul backend de stockage Azure NetApp Files ou un pool virtuel de backends de stockage Azure NetApp Files.

Avant de commencer

- Assurez-vous d'avoir activé Trident sur le cluster Red Hat OpenShift géré.
- Assurez-vous d'avoir accès à l' `tridentctl`utilitaire.
- Assurez-vous d'avoir préparé le stockage Azure NetApp Files pour le chiffrement Kerberos en prenant note des exigences et en suivant les instructions dans "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 backend de stockage

Vous pouvez créer une configuration de stockage backend Azure NetApp Files qui inclut la capacité de chiffrement Kerberos.

À propos de cette tâche

Lorsque vous créez un fichier de configuration de stockage qui configure le chiffrement Kerberos, vous pouvez le définir de sorte qu'il soit appliqué à l'un des deux niveaux possibles :

- Le **niveau du backend de stockage** utilisant le `spec.kerberos` champ
- Le **niveau de pool virtuel** utilisant le champ `spec.storage.kerberos`

Lorsque vous définissez la configuration au niveau du pool virtuel, le pool est sélectionné à l'aide de l'étiquette dans la classe de stockage.

À chaque niveau, vous pouvez spécifier l'une des trois versions différentes du chiffrement 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 vie privée)

Étapes

1. Sur le cluster géré, créez un fichier de configuration du backend de stockage en utilisant l'un des exemples suivants, selon l'endroit où vous devez définir le backend de stockage (au niveau du backend de stockage ou au niveau du pool virtuel). Remplacez les valeurs entre crochets <> avec les informations de votre environnement :

Exemple de niveau de backend 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 niveau 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 backend échoue, cela signifie qu'il y a un problème avec la configuration du backend. Vous pouvez consulter les journaux pour en 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 à nouveau la commande create.

Créer une classe de stockage

Vous pouvez créer une classe de stockage pour provisionner des volumes avec chiffrement Kerberos.

Étapes

1. Créez un objet Kubernetes StorageClass en utilisant 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. Créez 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 devriez voir une sortie similaire à la suivante :

```
NAME          PROVISIONER             AGE
sc-nfs        csi.trident.netapp.io   15h
```

Approvisionner des volumes

Après avoir créé un système de stockage et une classe de stockage, vous pouvez maintenant provisionner un volume. Pour obtenir des instructions, consultez "Provisionner un volume".

Protégez les applications avec Trident Protect

Découvrez Trident Protect

NetApp Trident Protect offre des capacités avancées de gestion des données applicatives qui améliorent la fonctionnalité et la disponibilité des applications Kubernetes avec état, soutenues 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 propose également des capacités d'automatisation via son API et son CLI.

Vous pouvez protéger les applications avec Trident Protect en créant des ressources personnalisées (CR) ou en utilisant la CLI Trident Protect.

Et ensuite ?

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 la conformité 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 exploiter Trident Protect.

Compatibilité de Trident Protect avec les clusters Kubernetes

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
- VMware Tanzu Portfolio
- 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 "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 backends de stockage suivants :

- Amazon FSx for NetApp ONTAP
- Cloud Volumes ONTAP
- baies de stockage ONTAP
- Google Cloud NetApp Volumes
- Azure NetApp Files

Assurez-vous que votre stockage backend réponde 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 NetApp ONTAP stockage backend.
- Assurez-vous d'avoir configuré un compartiment de stockage d'objets pour stocker les 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.

Exigences pour les volumes nas-economy

Trident Protect prend en charge les opérations de sauvegarde et de restauration sur les volumes nas-economy. Les snapshots, les clones et la réplication SnapMirror vers les 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 utilisant un répertoire de snapshots. Pour ces applications, vous devez masquer le répertoire de 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 de snapshots en exécutant la commande suivante pour chaque volume nas-economy, en remplaçant `<volume-UUID>` par l'UUID du volume que vous souhaitez modifier :

```
tridentctl update volume <volume-UUID> --snapshot-dir=true --pool-level
=true -n trident
```

REMARQUE	Vous pouvez activer par défaut les répertoires de snapshots pour les nouveaux volumes en définissant l'option de configuration du backend Trident `snapshotDir` sur `true`. Les volumes existants ne sont pas affectés.

Protection des 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 des machines virtuelles varient selon les versions de Trident Protect, les versions plus récentes proposant une configuration simplifiée via les paramètres du chart Helm.

REMARQUE	Lors des opérations de restauration, tout `VirtualMachineSnapshots` créé pour une machine virtuelle (VM) n'est pas restauré.

Trident Protect 25.10 et versions ultérieures

Trident Protect gèle et dégèle automatiquement les systèmes de fichiers KubeVirt lors des 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 du paramètre `vm.freeze` lors de l'installation du chart Helm. Le 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 dégèle 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 n'assure pas automatiquement un état cohérent pour les systèmes de fichiers KubeVirt des machines virtuelles lors des opérations de protection des données. Si vous souhaitez protéger les données de vos machines virtuelles KubeVirt avec Trident Protect 24.10, vous devez activer manuellement la fonctionnalité de gel/dégel pour les 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 en "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
```

Exigences pour la réplication SnapMirror

NetApp SnapMirror replication 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 for NetApp ONTAP

Exigences du cluster ONTAP pour la réplication SnapMirror

Assurez-vous que votre cluster ONTAP réponde aux exigences suivantes si vous prévoyez d'utiliser la réplication SnapMirror :

- **NetApp Trident** : NetApp Trident doit exister 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 en utilisant des classes de stockage reposant sur les pilotes suivants :
  - `ontap-nas` : NFS
  - `ontap-san` : iSCSI
  - `ontap-san` : FC
  - `ontap-san`: NVMe/TCP (nécessite la version minimale ONTAP 9.15.1)
- **Licences** : Les licences asynchrones ONTAP SnapMirror utilisant le Data Protection bundle doivent être activées sur les clusters ONTAP source et de destination. Consultez "SnapMirror aperçu des licences dans ONTAP" pour plus d'informations.

  À compter de ONTAP 9.10.1, toutes les licences sont fournies sous forme de fichier de licence NetApp (NLF), qui est un fichier unique permettant d'activer plusieurs fonctionnalités. Consultez "Licences incluses avec ONTAP One" pour plus d'informations.

REMARQUE	Seule la protection asynchrone SnapMirror est prise en charge.

Considérations de peering pour la réplication SnapMirror

Assurez-vous que votre environnement répond aux exigences suivantes si vous prévoyez d'utiliser le peering de stockage backend :

- **Cluster et SVM** : Les backends de stockage ONTAP doivent être appariés. Consultez "Aperçu du peering de cluster et de SVM" pour plus d'informations.

REMARQUE	Assurez-vous que les noms SVM utilisés dans la relation de réplication entre deux clusters ONTAP sont uniques.

- **NetApp Trident et SVM** : Les SVM distants appariés doivent être disponibles pour NetApp Trident sur le cluster de destination.
- **Systèmes de stockage backend gérés** : Vous devez ajouter et gérer des backends de stockage ONTAP dans Trident Protect pour créer une relation de réplication.

Configuration de Trident / ONTAP pour la réplication SnapMirror

Trident Protect exige que vous configuriez au moins un système de stockage 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 d'un cluster Kubernetes pour la réplication SnapMirror

Assurez-vous que vos clusters Kubernetes répondent aux exigences suivantes :

- **AppVault accessibilité** : Les clusters source et de destination doivent avoir un accès réseau pour lire depuis et écrire vers le 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 à travers les WAN.

REMARQUE	De nombreux environnements d'entreprise appliquent 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 depuis votre propre registre privé. L'installation depuis un registre privé est utile si votre cluster ne peut pas accéder à Internet.

Installer Trident Protect

Installez Trident Protect depuis NetApp

Étapes

1. Ajoutez le dépôt Trident Helm :

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

2. Utilisez Helm pour installer Trident Protect. Remplacez `<name-of-cluster>` par le nom du cluster, qui sera attribué au cluster et utilisé pour identifier les sauvegardes et instantanés du cluster :

```
helm install trident-protect netapp-trident-protect/trident-protect
--set clusterName=<name-of-cluster> --version 100.2602.0 --create
-namespace --namespace trident-protect
```

3. Facultativement, pour activer la journalisation de débogage (recommandée pour le dépannage), utilisez :

```
helm install trident-protect netapp-trident-protect/trident-protect
--set clusterName=<name-of-cluster> --set logLevel=debug --version
100.2602.0 --create-namespace --namespace trident-protect
```

La journalisation de débogage aide le support NetApp à résoudre les problèmes sans nécessiter de modification 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 de votre environnement :

Étapes

1. Téléchargez les images suivantes sur votre machine locale, mettez à jour les tags, puis poussez-les vers votre registre privé :

```
docker.io/netapp/controller:26.02.0
docker.io/netapp/restic:26.02.0
docker.io/netapp/kopia:26.02.0
docker.io/netapp/kopiablockrestore:26.02.0
docker.io/netapp/trident-autosupport:26.02.0
docker.io/netapp/exechook:26.02.0
docker.io/netapp/resourcebackup:26.02.0
docker.io/netapp/resourcerestore:26.02.0
docker.io/netapp/resourcedelete:26.02.0
docker.io/netapp/trident-protect-utils:v1.0.0
```

Par exemple :

```
docker pull docker.io/netapp/controller:26.02.0
```

```
docker tag docker.io/netapp/controller:26.02.0 <private-registry-url>/controller:26.02.0
```

```
docker push <private-registry-url>/controller:26.02.0
```

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 100.2602.0 --repo https://netapp.github.io/trident-protect-helm-chart`, puis copiez le fichier `trident-protect-100.2602.0.tgz` résultant dans votre environnement hors ligne et installez-le en utilisant `helm install trident-protect ./trident-protect-100.2602.0.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 d'extraction à utiliser pour l'authentification du 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. Ajoutez le dépôt 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 Les `imageRegistry` et `imagePullSecrets` 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 dans votre registre (par exemple, `example.com:443/my-repo`), indiquez le chemin complet dans le champ du registre. Cela garantira que toutes les images sont extraites de `<private-registry-url>/<image-name>:<tag>`.

7. Utilisez Helm pour installer Trident Protect. Remplacez `<name_of_cluster>` par le nom du cluster, qui sera attribué au cluster et utilisé pour identifier les sauvegardes et instantanés du cluster :

```
helm install trident-protect netapp-trident-protect/trident-protect
--set clusterName=<name_of_cluster> --version 100.2602.0 --create
-namespace --namespace trident-protect -f protectValues.yaml
```

8. Facultativement, pour activer la journalisation de débogage (recommandée pour le dépannage), utilisez :

```
helm install trident-protect netapp-trident-protect/trident-protect
--set clusterName=<name-of-cluster> --set logLevel=debug --version
100.2602.0 --create-namespace --namespace trident-protect -f
protectValues.yaml
```

La journalisation de débogage aide le support NetApp à résoudre les problèmes sans nécessiter de modification du niveau de journalisation ni de reproduction du problème.

REMARQUE	Pour plus d'options de configuration du chart Helm, y compris les paramètres AutoSupport et le filtrage des espaces de noms, reportez-vous à "Personnaliser l'installation de Trident Protect".

Installez le plugin CLI Trident Protect

Vous pouvez utiliser le plugin de ligne de commandes Trident Protect, qui est une extension de l'utilitaire Trident `tridentctl`, pour créer et interagir avec les ressources personnalisées (CR) Trident Protect.

Installez le plugin CLI Trident Protect

Avant d'utiliser l'utilitaire de ligne de commandes, vous devez l'installer sur la machine que vous utilisez pour accéder à votre cluster. Suivez ces étapes, selon que votre machine utilise un processeur x64 ou ARM.

Télécharger le plugin 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/26.02.0/tridentctl-protect-linux-amd64
```

Télécharger le plugin 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/26.02.0/tridentctl-protect-linux-arm64
```

Télécharger le plugin 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/26.02.0/tridentctl-protect-macos-amd64
```

Télécharger le plugin 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/26.02.0/tridentctl-protect-macos-arm64
```

1. Activer les permissions d'exécution pour le fichier binaire du plugin :

```
chmod +x tridentctl-protect
```

2. Copiez le fichier binaire du plugin dans un emplacement défini dans votre variable PATH. Par exemple, `/usr/bin` ou `/usr/local/bin` (des privilèges élevés peuvent être nécessaires) :

```
cp ./tridentctl-protect /usr/local/bin/
```

3. Vous pouvez également copier le fichier binaire du plugin dans un emplacement de votre répertoire local. Dans ce cas, il est recommandé de s'assurer que cet emplacement figure dans votre variable PATH :

```
cp ./tridentctl-protect ~/bin/
```

REMARQUE Copier le plugin dans un emplacement figurant dans votre variable PATH vous permet de l'utiliser en tapant `tridentctl-protect` ou `tridentctl protect` depuis n'importe quel emplacement.

Afficher l'aide du plugin Trident CLI

Vous pouvez utiliser les fonctionnalités d'aide intégrées du plugin pour obtenir une aide détaillée sur les capacités du plugin :

Étapes

1. Utilisez la fonction d'aide pour consulter les instructions d'utilisation :

```
tridentctl-protect help
```

Activer la saisie semi-automatique des commandes

Après avoir installé le plugin CLI Trident Protect, 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 local pour y placer le script :

```
mkdir -p ~/.bash/completions
```

3. Déplacez le script téléchargé dans le répertoire `~/.bash/completions` :

```
mv tridentctl-completion.bash ~/.bash/completions/
```

4. Ajoutez la ligne suivante au fichier `~/.bashrc` dans votre répertoire local :

```
source ~/.bash/completions/tridentctl-completion.bash
```

Activer la saisie semi-automatique pour le Z shell

Étapes

1. Créez le script de complétion :

```
tridentctl-protect completion zsh > tridentctl-completion.zsh
```

2. Créez un nouveau répertoire dans votre répertoire local pour y placer le script :

```
mkdir -p ~/.zsh/completions
```

3. Déplacez le script téléchargé dans le répertoire `~/.zsh/completions` :

```
mv tridentctl-completion.zsh ~/.zsh/completions/
```

4. Ajoutez la ligne suivante au fichier `~/.zprofile` dans votre répertoire local :

```
source ~/.zsh/completions/tridentctl-completion.zsh
```

Résultat

Lors de votre prochaine connexion à l'interpréteur de commandes, vous pourrez utiliser la saisie semi-automatique des commandes 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 pour les conteneurs Trident Protect après avoir installé Trident Protect. La définition de limites de ressources vous permet de contrôler la quantité de ressources du cluster consommée 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 présente les paramètres disponibles et contient les valeurs par défaut pour chaque limite de ressources :

```
---
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. Appliquez les valeurs du fichier `resourceLimits.yaml` :

```
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 (SCCs) pour les conteneurs Trident Protect après avoir installé Trident Protect. Ces contraintes définissent les restrictions de sécurité pour les 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 selon les besoins de votre environnement.

   L'exemple suivant montre les valeurs par défaut des paramètres pour l'option SCC :

```
scc:
  create: true
  name: trident-protect-job
  priority: 1
```

Ce tableau décrit les paramètres de l'option SCC :

Paramètre	Description	Défaut
créer	Détermine si une ressource SCC peut être créée. Une ressource SCC sera créée uniquement si `scc.create` est définie sur `true` et si le processus d'installation Helm identifie un environnement OpenShift. Si ce n'est pas sur OpenShift, ou si `scc.create` est définie sur `false`, aucune ressource SCC ne sera créée.	true
nom	Spécifie le nom du SCC.	trident-protect-job
priorité	Définit la priorité du SCC. Les SCC ayant une valeur de priorité plus élevée sont évalués avant ceux ayant une valeur plus faible.	1

3. Appliquez les valeurs du fichier `sccconfig.yaml` :

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect -f sccconfig.yaml --reuse-values
```

Cela remplacera les valeurs par défaut par celles spécifiées dans le `sccconfig.yaml` fichier.

Configurer des paramètres supplémentaires du chart Helm Trident Protect

Vous pouvez personnaliser les paramètres AutoSupport et le filtrage des espaces de noms pour répondre à vos besoins spécifiques. Le tableau suivant décrit les paramètres de configuration disponibles :

Paramètre	Type	Description
autoSupport.proxy	chaîne	Configure une URL de proxy pour NetApp AutoSupport connexions. Utilisez ceci pour acheminer les chargements de bundles de support via un serveur proxy. Exemple : `http://my.proxy.url`.
autoSupport.insecure	booléen	Désactive la vérification TLS pour AutoSupport proxy connections lorsque défini sur `true`. À utiliser uniquement pour les connexions proxy non sécurisées. (par défaut : `false`)

Paramètre	Type	Description
AutoSupport.enabled	booléen	Active ou désactive les chargements quotidiens de bundles Trident Protect AutoSupport. Lorsqu'elle est définie sur `false`, les chargements quotidiens planifiés sont désactivés, mais vous pouvez toujours générer manuellement des bundles de support. (par défaut : `true`)
restoreSkipNamespaceAnnotations	chaîne	Liste d'annotations d'espace de noms, séparées par des virgules, à exclure des opérations de sauvegarde et de restauration. Permet de filtrer les espaces de noms en fonction des annotations.
restoreSkipNamespaceLabels	chaîne	Liste séparée par des virgules d'étiquettes d'espace de noms à exclure des opérations de sauvegarde et de restauration. Permet de filtrer les espaces de noms en fonction des étiquettes.

Vous pouvez configurer ces options à l'aide d'un fichier de configuration YAML ou de paramètres de ligne de commandes :

Utiliser un 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"
```

3. Après avoir rempli le fichier `values.yaml` 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
```

Utilisez le paramètre CLI

Étapes

1. Utilisez la commande suivante avec le `--set` paramètre 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}" \
  --reuse-values
```

Limiter les pods Trident Protect à des nœuds spécifiques

Vous pouvez utiliser la contrainte de sélection de nœuds Kubernetes nodeSelector pour contrôler quels nœuds sont autorisés à 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 selon 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 les étiquettes des nœuds afin de les restreindre selon les besoins de votre environnement. Par exemple, le fichier suivant contient la restriction OS par défaut, mais cible également une région spécifique et un nom d'application :

```
nodeSelector:
  kubernetes.io/os: linux
  region: us-west
  app.kubernetes.io/name: mysql
```

3. Appliquez les valeurs du fichier `nodeSelectorConfig.yaml` :

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect -f nodeSelectorConfig.yaml --reuse-values
```

Cela remplace les restrictions par défaut par celles que vous avez spécifiées dans le `nodeSelectorConfig.yaml` fichier.

Gérer Trident Protect

Gérer l'autorisation Trident et le contrôle d'accès

Trident Protect utilise le modèle Kubernetes de contrôle d'accès basé sur les rôles (RBAC). Par défaut, Trident Protect fournit un espace de noms système unique et son compte de service par défaut associé. Si vous avez une organisation avec de nombreux utilisateurs ou des besoins de sécurité spécifiques, vous pouvez utiliser les fonctionnalités RBAC de Trident Protect pour obtenir un contrôle plus granulaire sur l'accès aux ressources et aux espaces de noms.

L'administrateur du cluster a toujours accès aux ressources dans l' `trident-protect`namespace par défaut, et peut également accéder aux ressources dans tous les autres namespaces. Pour contrôler l'accès aux ressources et aux applications, vous devez créer des namespaces supplémentaires et ajouter des ressources et des applications à ces namespaces.

Notez qu'aucun utilisateur ne peut créer des CR de gestion des données d'application dans l' `trident-protect`espace de noms par défaut. Vous devez créer des CR de gestion des données d'application dans un espace de noms d'application (comme bonne pratique, créez les CR de gestion des données d'application dans le même espace de noms que leur application associée).

REMARQUE	Seuls les administrateurs devraient avoir accès aux objets de ressources personnalisés privilégiés de Trident Protect, notamment : • **AppVault** : Nécessite des identifiants de compartiment • **AutoSupportBundle** : Collecte des métriques, des journaux et d'autres données sensibles Trident Protect • **AutoSupportBundleSchedule**: Gère les calendriers de collecte des journaux En tant que bonne pratique, utilisez le RBAC pour limiter l'accès aux objets privilégiés aux administrateurs.

Pour plus d'informations sur la manière dont le RBAC régule l'accès aux ressources et aux espaces de noms, reportez-vous à la "Documentation RBAC Kubernetes".

Pour plus d'informations sur les comptes de service, veuillez vous référer au "Documentation du compte de service Kubernetes".

Exemple : gérer l'accès pour deux groupes d'utilisateurs

Par exemple, une organisation dispose d'un administrateur du cluster, d'un groupe d'utilisateurs ingénieurs et d'un groupe d'utilisateurs marketing. L'administrateur du cluster effectuerait les tâches suivantes pour créer un environnement où le groupe d'ingénierie et le groupe marketing ont chacun accès uniquement aux ressources attribuées à leurs espaces de noms respectifs.

Étape 1 : Créer un espace de noms pour contenir les ressources de 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 noms pour le groupe d'ingénierie :

```
kubectl create ns engineering-ns
```

2. Créez un espace de noms pour le groupe marketing :

```
kubectl create ns marketing-ns
```

Étape 2 : Créez de nouveaux comptes de service pour interagir avec les ressources dans 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 les privilèges entre les groupes à l'avenir 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. Créer 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 service account

Un secret de compte de service est utilisé pour s'authentifier avec le compte de service et peut facilement être supprimé et recréé en cas de compromission.

Étapes

1. Créez un secret pour le compte du 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 du 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 le 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. Liez le 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 : Tester les autorisations

Testez que les autorisations sont correctes.

Étapes

1. Confirmez que les utilisateurs ingénieurs 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. Confirmez que les utilisateurs ingénieurs 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 instantanés, l'administrateur du cluster doit accorder l'accès aux objets AppVault à chaque utilisateur.

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, le CR suivant 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 un Role CR pour permettre aux administrateurs du 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 un 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 devriez voir une sortie similaire à la suivante :

```
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. Vérifiez si l'utilisateur peut obtenir les informations AppVault auxquelles il a désormais accès :

```
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 devriez voir une sortie similaire à la suivante :

```
yes
```

Résultat

Les utilisateurs auxquels vous avez accordé des autorisations AppVault doivent pouvoir utiliser les objets AppVault autorisés pour les opérations de gestion des données d'application, et ne doivent pas pouvoir accéder à des ressources en dehors des espaces de noms attribués ni 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 la santé des ressources protégées par Trident Protect.

Le service kube-state-metrics génère des métriques à partir des communications avec l'API Kubernetes. L'utiliser avec Trident Protect expose des informations utiles sur l'état des ressources dans votre environnement.

Prometheus est un ensemble d'outils permettant 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 fournissent un moyen de surveiller la santé et l'état 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 achemine vers des destinations que vous configurez.

REMARQUE

Les configurations et les conseils fournis dans ces étapes ne sont que des exemples ; vous devez les adapter à votre environnement. Consultez la documentation officielle suivante pour obtenir des instructions et une assistance spécifiques :

- "documentation kube-state-metrics"
- "Documentation Prometheus"
- "Documentation d'Alertmanager"

Étape 1 : Installer les outils de surveillance

Pour activer la surveillance des ressources dans Trident Protect, vous devez installer et configurer kube-state-metrics, Prometheus, et Alertmanager.

Installer kube-state-metrics

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 ServiceMonitor Prometheus 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 Helm chart (par exemple, `metrics-config.yaml`). Vous pouvez personnaliser l'exemple de configuration suivant pour l'adapter à votre environnement :

metrics-config.yaml : configuration du Helm chart 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 kube-state-metrics en déployant le chart 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 dans le "Documentation sur la ressource personnalisée kube-state-metrics".

Installer Prometheus

Vous pouvez installer Prometheus en suivant les instructions dans le "Documentation Prometheus".

Installer Alertmanager

Vous pouvez installer Alertmanager en suivant les instructions dans le "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 kube-state-metrics à 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 métriques depuis kube-state-metrics et peut envoyer des alertes à Alertmanager. Vous êtes maintenant prêt à configurer les conditions qui déclenchent une alerte et où les alertes doivent être envoyées.

Étape 3 : Configurer les alertes et les destinations d'alerte

Après avoir configuré les outils pour fonctionner ensemble, vous devez configurer le type d'informations qui déclenchent les alertes et l'endroit où les alertes doivent être envoyées.

Exemple d'alerte : échec de la sauvegarde

L'exemple suivant définit une alerte critique qui se déclenche 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 fichier de configuration `prometheus.yaml` :

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 vers d'autres canaux

Vous pouvez configurer Alertmanager pour envoyer des notifications à d'autres canaux, tels que le courrier électronique, PagerDuty, Microsoft Teams ou d'autres services de notification, en spécifiant la configuration correspondante dans le `alertmanager.yaml` fichier.

L'exemple suivant configure Alertmanager pour envoyer des notifications à un canal Slack. Pour adapter cet exemple à votre environnement, remplacez la valeur de la `api_url` clé par l'URL du webhook Slack que vous utilisez :

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 bundle de support Trident Protect

Trident Protect permet aux administrateurs de générer des bundles contenant des informations utiles à NetApp Support, notamment les journaux, les métriques et les informations de topologie des clusters et des applications gérés. Si vous êtes connecté à Internet, vous pouvez charger des bundles de support sur le NetApp Support Site (NSS) à l'aide d'un fichier de ressource personnalisé (CR).

Créez un bundle de support à l'aide d'un 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**: (*Obligatoire*) Le nom de cette ressource personnalisée; choisissez un nom unique et pertinent pour votre environnement.
   - **spec.triggerType** : (*Obligatoire*) Détermine si le support bundle est généré immédiatement ou planifié. La génération planifiée du support bundle a lieu à 12h00 UTC. Valeurs possibles :
     - Programmé
     - Manuel
   - **spec.uploadEnabled** : (*Facultatif*) Contrôle si le bundle de support doit être téléchargé sur le site de support NetApp après sa génération. Si non spécifié, la valeur par défaut est `false`. Valeurs possibles :
     - true
     - faux (default)
   - **spec.dataWindowStart** : (*Facultatif*) Une chaîne de date au format RFC 3339 qui spécifie la date et l'heure auxquelles la fenêtre des données incluses dans le bundle de support doit commencer. Si elle n'est pas spécifiée, la valeur par défaut est il y a 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 fichier `trident-protect-support-bundle.yaml` 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 la ligne de commandes

Étapes

1. Créez le bundle de support en remplaçant les valeurs entre crochets par les informations de votre

environnement. Le `trigger-type` détermine si le bundle est créé immédiatement ou si l'heure de création est dictée par la planification, 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 bundle de support

Après avoir créé un bundle de support en utilisant l'une ou l'autre méthode, vous pouvez suivre la progression de sa génération et le récupérer sur votre système local.

Étapes

1. Attendez que le `status.generationState` atteigne l' `Completed` état. Vous pouvez suivre la progression de la génération avec la commande suivante :

   ```
   kubectl get autosupportbundle trident-protect-support-bundle -n trident-
   protect
   ```

2. Récupérez le support bundle 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
   ```

   Trouvez la commande `kubectl cp` dans le résultat et exécutez-la en remplaçant l'argument de destination par votre répertoire local préféré.

Mettre à niveau Trident Protect

Vous pouvez mettre à jour Trident Protect vers la dernière version pour bénéficier de nouvelles fonctionnalités ou de correctifs de bugs.

REMARQUE

- Lors de la mise à niveau depuis la version 24.10, les instantanés en cours d'exécution pendant la mise à niveau peuvent échouer. Cet échec n'empêche pas la création future d'instantanés, qu'ils soient manuels ou planifiés. Si un instantané échoue pendant la mise à niveau, vous pouvez créer un nouvel instantané manuellement pour garantir la protection de votre application.

  Pour éviter d'éventuels dysfonctionnements, vous pouvez désactiver toutes les planifications de snapshots avant la mise à niveau et les réactiver ensuite. Cependant, cela entraîne l'absence de snapshots planifiés pendant la période de mise à niveau.

- Pour les installations de registre privé, assurez-vous que le Helm chart 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 chart. Pour plus d'informations, consultez "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, procédez à 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 entre dans 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. Mettre à jour le dépôt 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 chart Helm Trident Protect.

a. Exécutez cette commande pour transférer la gestion des CRD de `trident-protect-crds` vers `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` chart :

REMARQUE Ne désinstallez pas le `trident-protect-crds`chart à l'aide de Helm, car cela pourrait supprimer vos CRDs 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 100.2602.0 --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`. La journalisation de débogage est recommandée pour le dépannage, car elle aide NetApp support à diagnostiquer les problèmes sans nécessiter de modification du niveau de journalisation ni de reproduction du problème.

Gérer et protéger les applications

Utilisez les objets Trident Protect AppVault pour gérer les compartiments

La ressource personnalisée (CR) de compartiment pour Trident Protect est appelée un AppVault. Les objets AppVault sont la représentation déclarative du workflow Kubernetes d'un bucket de stockage. Une CR AppVault contient les configurations nécessaires pour qu'un bucket soit utilisé dans les opérations de protection, telles que les sauvegardes, les instantanés, les opérations de restauration et la réplication SnapMirror. Seuls les administrateurs peuvent créer des AppVaults.

Vous devez créer une AppVault CR manuellement ou à partir de la ligne de commandes lorsque vous effectuez des opérations de protection des données sur une application. Le AppVault CR est spécifique à votre environnement, et vous pouvez utiliser les exemples de cette page comme guide lors de la création de AppVault CR.

REMARQUE	Assurez-vous que le AppVault CR se trouve sur le cluster où Trident Protect est installé. Si le AppVault CR n'existe pas ou si vous ne pouvez pas y accéder, la ligne de commandes affiche une erreur.

Configurer l'authentification et les mots de passe AppVault

Avant de créer un AppVault CR, assurez-vous que le AppVault et le data mover 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 Trident Protect CLI, vous pouvez spécifier un secret Kubernetes avec des mots de passe personnalisés pour le chiffrement Restic et Kopia. Si vous ne spécifiez pas de 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 la ligne de commandes Trident Protect, utilisez l'argument `--data-mover-password-secret-ref` pour spécifier le secret.

Créer un secret de mot de passe pour le référentiel de déplacement de données

Utilisez les exemples suivants pour créer le mot de passe secret. Lorsque vous créez des objets AppVault, vous pouvez demander à Trident Protect d'utiliser ce secret pour s'authentifier auprès du référentiel du data mover.

REMARQUE	• Selon le data mover que vous utilisez, il vous suffit d'inclure le mot de passe correspondant pour ce data mover. Par exemple, si vous utilisez Restic et que vous ne prévoyez pas d'utiliser Kopia à l'avenir, vous pouvez inclure uniquement le mot de passe Restic lors de la création du secret. • Conservez le mot de passe en lieu sûr. Vous en aurez besoin pour restaurer les données sur le même cluster ou un autre. Si le cluster ou le `trident-protect` namespace est supprimé, vous ne pourrez pas restaurer vos sauvegardes ou instantanés sans le mot de passe.

Utilisez un 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
```

Utilisez la ligne de commandes (CLI)

```
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, S3 générique, "StorageGrid S3", ou "ONTAP S3" en utilisant Trident Protect, vous devez vous assurer que les informations d'identification de l'utilisateur disposent des autorisations nécessaires pour accéder au compartiment. Voici un exemple de stratégie accordant les autorisations minimales requises pour l'accès avec Trident Protect. Vous pouvez appliquer cette stratégie à l'utilisateur qui gère les stratégies des compartiments 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".

EKS Pod Identity 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 compartiments S3 sans stocker les informations d'identification AWS dans les secrets Kubernetes.

Exigences pour EKS Pod Identity 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 disposant des autorisations nécessaires pour le compartiment S3. 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 Pod Identity et l'association des rôles IAM aux comptes de service, reportez-vous à la "Documentation AWS EKS Pod Identity".

AppVault Configuration Lors de l'utilisation d'EKS Pod Identity, configurez votre AppVault CR avec le `useIAM: true` indicateur 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
```

AppVault exemples de génération de clés pour les fournisseurs de cloud

Lors de la définition d'une AppVault CR, vous devez inclure les informations d'identification permettant d'accéder aux ressources hébergées par le fournisseur, sauf si vous utilisez l'authentification IAM. La façon dont vous générez les clés pour les informations d'identification varie selon le fournisseur. Voici des exemples de génération de clés en ligne de commandes pour plusieurs fournisseurs. Vous pouvez utiliser les exemples suivants pour créer les 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
```

AppVault creation exemples

Les exemples suivants sont des définitions AppVault pour chaque fournisseur.

AppVault CR exemples

Vous pouvez utiliser les exemples CR suivants pour créer des objets AppVault pour chaque fournisseur de cloud.

REMARQUE

- Vous pouvez spécifier, si vous le souhaitez, un secret Kubernetes contenant des mots de passe personnalisés pour le chiffrement du référentiel Restic et Kopia. Consultez Mots de passe du référentiel de data mover pour plus d'informations.
- Pour les objets AppVault Amazon S3 (AWS), vous pouvez éventuellement spécifier un sessionToken, ce qui est utile si vous utilisez l'authentification par connexion unique (SSO). Ce jeton est créé lorsque vous générez des clés pour le fournisseur dans AppVault exemples de génération de clés pour les fournisseurs de cloud.
- Pour les objets S3 AppVault, vous pouvez éventuellement spécifier une URL de proxy de sortie pour le trafic S3 sortant à l'aide de la clé `spec.providerConfig.S3.proxyURL`.

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 Kopia data mover, vous pouvez supprimer la `providerCredentials` section et ajouter `useIAM: true` sous la configuration `s3` 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
```

AppVault Exemples de création à l'aide de la ligne de commandes Trident Protect

Vous pouvez utiliser les exemples de commandes CLI suivants pour créer des CR AppVault pour chaque fournisseur.

REMARQUE

- Vous pouvez spécifier, si vous le souhaitez, un secret Kubernetes contenant des mots de passe personnalisés pour le chiffrement du référentiel Restic et Kopia. Consultez Mots de passe du référentiel de data mover pour plus d'informations.
- Pour les objets S3 AppVault, vous pouvez éventuellement spécifier une URL de proxy de sortie pour le trafic S3 sortant à l'aide de l'argument `--proxy-url <ip_address:port>`.

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
```

Options de configuration `providerConfig.s3` prises en charge

Consultez le tableau suivant pour connaître les options de configuration du fournisseur S3 :

Paramètre	Description	Défaut	Exemple
`providerConfig.s3.skipCertValidation`	Désactivez la vérification du certificat SSL/TLS.	false	"true", "false"
`providerConfig.s3.secure`	Activez la communication HTTPS sécurisée avec le point de terminaison S3.	true	"true", "false"
`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 EKS Pod Identity.	false	vrai, faux

Afficher les informations AppVault

Vous pouvez utiliser le plugin CLI Trident Protect pour consulter les informations sur les 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 résultat:

```
+-------------+-------+----------+----------------------------
+---------------------------+
|   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. Pour afficher éventuellement le AppVaultPath pour chaque ressource, utilisez le drapeau `--show-paths`.

   Le nom du cluster dans la première colonne du tableau n'est disponible que si un nom de cluster a été spécifié lors de l'installation Helm de Trident Protect. Par exemple : `--set clusterName=production1`.

Retirer un AppVault

Vous pouvez supprimer un AppVault objet à tout moment.

REMARQUE	Ne supprimez pas la `finalizers` clé dans le AppVault CR avant de supprimer l'objet AppVault. Si vous le faites, cela peut entraîner des données résiduelles dans le compartiment AppVault et des ressources orphelines dans le cluster.

Avant de commencer

Assurez-vous d'avoir supprimé toutes les CR de snapshot et de sauvegarde utilisées par le AppVault que vous souhaitez supprimer.

Supprimez un AppVault à l'aide de la CLI Kubernetes

1. Supprimez l'objet AppVault, en remplaçant `appvault-name` par le nom de l'objet AppVault à supprimer :

```
kubectl delete appvault <appvault-name> \
-n trident-protect
```

Supprimez un AppVault à l'aide de la ligne de commandes Trident Protect

1. Supprimez l'objet AppVault, en remplaçant `appvault-name` par le nom de l'objet AppVault à supprimer :

```
tridentctl-protect delete appvault <appvault-name> \
-n trident-protect
```

Définir une application pour la gestion avec Trident Protect

Vous pouvez définir une application que vous souhaitez gérer avec Trident Protect en créant un CR d'application et un CR AppVault associé.

Créer un AppVault CR

Vous devez créer une AppVault CR qui sera utilisée lors des opérations de protection des données sur l'application, et la AppVault CR doit résider sur le cluster où Trident Protect est installé. La AppVault CR est spécifique à votre environnement ; pour des exemples de AppVault CR, consultez "AppVault ressources personnalisées."

Définir une application

Vous devez définir chaque application que vous souhaitez gérer avec Trident Protect. Vous pouvez définir une application à gérer soit en créant manuellement un CR d'application, soit en utilisant le CLI de Trident Protect.

Ajouter une application à l'aide d'un CR

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** : (*Obligatoire*) Le nom de la ressource personnalisée de l'application. Notez le nom que vous choisissez, car d'autres fichiers CR nécessaires aux opérations de protection font référence à cette valeur.
      - **spec.includedNamespaces** : (*Obligatoire*) Utilisez un espace de noms et un sélecteur d'étiquette pour spécifier les espaces de noms et les ressources utilisés par l'application. L'espace de noms de l'application doit figurer dans cette liste. Le sélecteur d'étiquette est facultatif et peut être utilisé pour filtrer les ressources au sein de chaque espace de noms spécifié.
      - **spec.includedClusterScopedResources**: (*Facultatif*) Utilisez cet attribut pour spécifier les ressources de portée cluster à inclure dans la définition de l'application. Cet attribut vous permet de sélectionner ces ressources en fonction de leur groupe, version, type et étiquettes.
        - **groupVersionKind**: (*Obligatoire*) Spécifie le groupe d'API, la version et le type de la ressource à portée de cluster.
        - **labelSelector** : (*Optionnel*) Filtre les ressources à portée du cluster en fonction de leurs étiquettes.
      - **metadata.annotations.protect.trident.netapp.io/skip-vm-freeze** : (*Optionnel*) Cette annotation s'applique uniquement aux applications définies à partir de machines virtuelles, telles que dans les environnements KubeVirt, où le gel du système de fichiers intervient avant la création d'instantanés. Indiquez si cette application peut écrire sur le système de fichiers pendant la création d'un instantané. Si la valeur est true, l'application ignore le paramètre global et peut écrire sur le système de fichiers pendant la création d'un instantané. Si la valeur est false, l'application ignore le paramètre global et le système de fichiers est gelé pendant 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. Si elle n'est pas spécifiée, l'application suit le "paramètre de gel global Trident Protect".

Si vous devez appliquer cette annotation après qu'une application a déjà été créée, 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** : (Obligatoire pour le filtrage) Utilisez `Include` ou `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 selon une opération OU, et les champs à l'intérieur de chaque élément (group, kind, version) correspondent selon une opération ET.
       - **resourceMatchers[].group**: (*Optionnel*) Groupe de la ressource à filtrer.
       - **resourceMatchers[].kind**: (*Optionnel*) Type de ressource à filtrer.

- **resourceMatchers[].version**: (*Optionnel*) Version de la ressource à filtrer.
- **resourceMatchers[].names**: (*Optionnel*) Noms dans le champ Kubernetes metadata.name de la ressource à filtrer.
- **resourceMatchers[].namespaces**: (*Optionnel*) Espaces de noms dans le champ metadata.name de Kubernetes 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"`.

REMARQUE	Lorsque `resourceFilter` et `labelSelector` sont utilisés, `resourceFilter` s'exécute en premier, 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 `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. Après avoir créé la CR d'application adaptée à votre environnement, appliquez la CR. Par exemple :

```
kubectl apply -f maria-app.yaml
```

Étapes

1. Créez et appliquez la définition d'application en utilisant l'un des exemples suivants, en remplaçant les valeurs entre crochets par les informations de votre environnement. Vous pouvez inclure des espaces de noms et des ressources dans la définition d'application en utilisant des listes séparées par des virgules avec les arguments indiqués dans les exemples.

   Vous pouvez éventuellement utiliser une annotation lors de la création d'une application pour spécifier si l'application peut écrire sur le système de fichiers pendant un instantané. Ceci s'applique uniquement aux applications définies à partir de machines virtuelles, comme dans les environnements KubeVirt, où le système de fichiers est gelé avant les instantanés. Si vous définissez l'annotation sur `true`, l'application ignore le paramètre global et peut écrire sur le système de fichiers pendant un instantané. Si vous la définissez sur `false`, l'application ignore le paramètre global et le système de fichiers est gelé pendant un instantané. Si vous utilisez l'annotation mais que l'application ne comporte aucune machine virtuelle dans la définition de l'application, l'annotation est ignorée. Si vous n'utilisez pas l'annotation, l'application suit le "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 de sauvegarde dépendent de `PersistentVolume` (le volume à portée du cluster lié à chaque PVC) et `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":"Persis
tentVolume","Version":"v1"},{"Kind":"VolumeSnapshotClass","Versio
n":"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

snapshots 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. "En savoir plus sur la configuration du gel du système de fichiers avec Trident Protect".

Créer un instantané à la demande

Vous pouvez créer un instantané à la demande à tout moment.

REMARQUE	Les ressources à portée de cluster sont incluses dans une sauvegarde, un instantané ou un clone si elles sont explicitement référencées dans la définition de l'application ou si elles ont des références à l'un des espaces de noms de l'application.

Créez 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**: (*Obligatoire*) Le nom de cette ressource personnalisée; choisissez un nom unique et pertinent pour votre environnement.
   - **spec.applicationRef** : Le nom Kubernetes de l'application à capturer.
   - **spec.appVaultRef** : (*Obligatoire*) Le nom de l'AppVault où le contenu de l'instantané (métadonnées) doit être stocké.
   - **spec.reclaimPolicy** : (Optionnel) Définit ce qui arrive à l'AppArchive d'un instantané lorsque le CR de l'instantané est supprimé. Cela signifie que même lorsqu'il est défini sur `Retain`, l'instantané sera supprimé. Options valides :
     - `Retain` (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. Après avoir rempli le fichier `trident-protect-snapshot-cr.yaml` avec les valeurs correctes, appliquez le CR :

```
kubectl apply -f trident-protect-snapshot-cr.yaml
```

Créez un instantané à l'aide de l'interface de ligne de commande (CLI)

Étapes

1. Créez l'instantané en remplaçant les valeurs entre crochets 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>
```

Créer une sauvegarde à la demande

Vous pouvez sauvegarder une app à tout moment.

REMARQUE	Les ressources à portée de cluster sont incluses dans une sauvegarde, un instantané ou un clone si elles sont explicitement référencées dans la définition de l'application ou si elles ont des références à l'un des espaces de noms de l'application.

Avant de commencer

Assurez-vous que la durée de validité du jeton de session AWS est suffisante pour toute opération de sauvegarde s3 de longue durée. Si le jeton expire pendant l'opération de sauvegarde, l'opération peut échouer.

- Consultez la "Documentation AWS API" pour plus d'informations sur la vérification de l'expiration du jeton de session actuel.
- Consultez la "Documentation AWS IAM" pour plus d'informations sur les identifiants relatifs aux ressources AWS.

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**: (*Obligatoire*) Le nom de cette ressource personnalisée; choisissez un nom unique et pertinent pour votre environnement.
   - **spec.applicationRef**: (*Obligatoire*) Le nom Kubernetes de l'application à sauvegarder.
   - **spec.appVaultRef** : (*Obligatoire*) Le nom du AppVault où le contenu de la sauvegarde doit être stocké.
   - **spec.dataMover** : (*Facultatif*) Chaîne de caractères indiquant l'outil de sauvegarde à utiliser pour l'opération de sauvegarde. Valeurs possibles (sensibles à la casse) :
     - `Restic`
     - `Kopia` (défaut)
   - **spec.reclaimPolicy** : (*Optionnel*) Définit ce qui se passe pour une sauvegarde lorsqu'elle est libérée de sa revendication. Valeurs possibles :
     - `Delete`
     - `Retain` (défaut)
   - **spec.snapshotRef** : (*Facultatif*) : Nom de l'instantané à utiliser comme source de la sauvegarde. Si cette information n'est pas fournie, 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. Après avoir rempli le fichier `trident-protect-backup-cr.yaml` avec les valeurs correctes, appliquez le CR :

```
kubectl apply -f trident-protect-backup-cr.yaml
```

Créez une sauvegarde à l'aide de l'interface de ligne de commande (CLI)

Étapes

1. Créez la sauvegarde en remplaçant les valeurs entre crochets 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 le `--full-backup` flag pour spécifier si une sauvegarde doit être non incrémentielle. Par défaut, toutes les sauvegardes sont incrémentielles. Lorsque ce flag est utilisé, la sauvegarde devient non incrémentielle. Il est bonne pratique d'effectuer une sauvegarde complète périodiquement, puis d'effectuer des sauvegardes incrémentielles entre les sauvegardes complètes afin de minimiser 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'un CR de sauvegarde :

Annotation	Type	Description	valeur par défaut
protect.trident.netapp.io/full-backup	chaîne	Indique si une sauvegarde doit être non incrémentielle. Définissez sur `true` pour créer une sauvegarde non incrémentielle. Il est bonne pratique d'effectuer une sauvegarde complète périodiquement, puis d'effectuer des sauvegardes incrémentielles entre les sauvegardes complètes afin de minimiser le risque associé aux restaurations.	"false"
protect.trident.netapp.io/snapshot-completion-timeout	chaîne	Le temps maximal autorisé pour que l'opération globale de capture d'instantané soit terminée.	"60m"
protect.trident.netapp.io/volume-snapshots-ready-to-use-timeout	chaîne	Le temps 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	Temps maximal (en secondes) d'attente pour que tout nouveau PersistentVolumeClaims (PVC) atteigne la phase `Bound` avant que l'opération n'échoue.	"1200" (20 minutes)

Créer un calendrier de protection des données

Une politique de protection protège une app en créant des instantanés, des sauvegardes, ou les deux selon une planification définie. 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 à l'aide de l'annotation full-backup-rule. Par défaut, toutes les sauvegardes sont incrémentielles. Effectuer une sauvegarde complète périodiquement,

ainsi que des sauvegardes incrémentielles entre les deux, aide à 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. Définir `snapshotRetention` à zéro signifie que toute sauvegarde planifiée créera toujours des instantanés, mais ceux-ci seront temporaires et supprimés immédiatement après la fin de la sauvegarde.
- Les ressources à portée de cluster sont incluses dans une sauvegarde, un instantané ou un clone si elles sont explicitement référencées dans la définition de l'application ou si elles ont des références à l'un des espaces de noms de l'application.

Créez un planning à l'aide d'une CR

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**: (*Obligatoire*) Le nom de cette ressource personnalisée; choisissez un nom unique et pertinent pour votre environnement.
   - **spec.dataMover** : (*Facultatif*) Chaîne de caractères indiquant l'outil de sauvegarde à utiliser pour l'opération de sauvegarde. Valeurs possibles (sensibles à la casse) :
     - `Restic`
     - `Kopia` (défaut)
   - **spec.applicationRef** : Le nom Kubernetes de l'application à sauvegarder.
   - **spec.appVaultRef** : (*Obligatoire*) Le nom du 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 se passe pour une sauvegarde si le CR de sauvegarde est supprimé pendant sa période de rétention. Après la période de rétention, les sauvegardes sont toujours supprimées. Valeurs possibles (sensibles à la casse) :
     - `Retain` (défaut)
     - `Delete`
   - **spec.snapshotRetention** : (*Obligatoire*) Le nombre d'instantanés à conserver. Zéro indique qu'aucun instantané ne doit être créé.
   - **spec.snapshotReclaimPolicy** : (*Optionnel*) Détermine ce qui se passe pour un instantané si la ressource personnalisée (CR) de l'instantané est supprimée pendant sa période de rétention. Après la période de rétention, les instantanés sont toujours supprimés. Valeurs possibles (sensibles à la casse) :
     - `Retain`
     - `Delete` (défaut)
   - **spec.granularity** : La fréquence à laquelle la planification doit s'exécuter. Valeurs possibles, ainsi que les champs associés obligatoires :
     - `Hourly` (nécessite que vous spécifiiez `spec.minute`)
     - `Daily` (nécessite que vous précisiez `spec.minute` et `spec.hour`)
     - `Weekly` (nécessite que vous précisiez `spec.minute, spec.hour`, et `spec.dayOfWeek`)
     - `Monthly` (nécessite que vous précisiez `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 requis 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) auquel la planification doit s'exécuter. Les valeurs 0 ou 7 indiquent le dimanche. Ce champ est requis 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 la planification 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 YAML pour la planification des sauvegardes et des instantanés :

```
---
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 YAML pour une planification uniquement par instantané :

```
---
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. Après avoir rempli le fichier `trident-protect-schedule-cr.yaml` avec les valeurs correctes, appliquez le CR :

```
kubectl apply -f trident-protect-schedule-cr.yaml
```

Créez une planification à l'aide de la CLI

Étapes

1. Créez le calendrier de protection en remplaçant les valeurs entre crochets par les informations de votre environnement. Par exemple :

REMARQUE Vous pouvez utiliser `tridentctl-protect create schedule --help` pour consulter des informations d'aide détaillées pour 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 offrent un contrôle supplémentaire sur votre emploi du temps :

- **Planification de sauvegarde complète** : Utilisez le `--full-backup-rule` indicateur pour planifier des sauvegardes complètes non incrémentielles. Cet indicateur fonctionne uniquement 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 : Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday.

    REMARQUE Le `--full-backup-rule`paramètre ne fonctionne pas avec la 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 d'instantané uniquement** : Définissez `--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'un schedule CR :

Annotation	Type	Description	valeur par défaut
protect.trident.netapp.io/full-backup-rule	chaîne	Spécifie la règle de planification des sauvegardes complètes. Vous pouvez la configurer sur `Always` pour une sauvegarde complète constante ou la personnaliser selon 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 valeurs valides pour les jours de la semaine sont : Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday. Notez que cette annotation ne peut être utilisée qu'avec les planifications qui ont `granularity` défini sur `Daily`.	Non défini (toutes les sauvegardes sont incrémentales)
protect.trident.netapp.io/snapshot-completion-timeout	chaîne	Le temps maximal autorisé pour que l'opération globale de capture d'instantané soit terminée.	"60m"
protect.trident.netapp.io/volume-snapshots-ready-to-use-timeout	chaîne	Le temps 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	Temps maximal (en secondes) d'attente pour que tout nouveau PersistentVolumeClaims (PVC) atteigne la phase `Bound` avant que l'opération n'échoue.	"1200" (20 minutes)

Supprimer un instantané

Supprimez les instantanés planifiés ou à la demande dont vous n'avez plus besoin.

Étapes

1. Supprimez le snapshot 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 dont vous n'avez plus besoin.

REMARQUE	Assurez-vous que la stratégie de récupération est configurée sur `Delete` pour supprimer toutes les données de sauvegarde du stockage objet. Le paramètre par défaut de la stratégie est `Retain` pour éviter toute perte de données. Si la stratégie n'est pas modifiée sur `Delete`, les données de sauvegarde resteront dans le stockage objet et devront être supprimées manuellement.

Étapes

1. Supprimez le CR de sauvegarde associé à la sauvegarde :

```
kubectl delete backup <backup_name> -n my-app-namespace
```

Vérifiez l'état d'une opération de sauvegarde

Vous pouvez utiliser la ligne de commandes pour vérifier l'état d'une opération de sauvegarde qui est en cours, terminée ou a échoué.

Étapes

1. Utilisez la commande suivante pour récupérer l'état de l'opération de sauvegarde, en remplaçant les valeurs entre crochets par les informations de votre environnement :

```
kubectl get backup -n <namespace_name> <my_backup_cr_name> -o jsonpath
='{.status}'
```

Activer la sauvegarde et la restauration pour les opérations azure-netapp-files (ANF)

Si vous avez installé Trident Protect, vous pouvez activer la fonctionnalité de sauvegarde et de restauration efficace en termes d'espace pour les systèmes de stockage utilisant la classe de stockage azure-netapp-files et créés avant Trident 24.06. Cette fonctionnalité fonctionne avec les volumes NFSv4 et ne consomme pas d'espace supplémentaire du pool de capacité.

Avant de commencer

Assurez-vous de ce qui suit :

- Vous avez installé Trident Protect.
- Vous avez défini une application dans Trident Protect. Cette application disposera d'une fonctionnalité de protection limitée jusqu'à ce que vous ayez terminé cette procédure.
- Vous avez `azure-netapp-files` sélectionné comme classe de stockage par défaut pour votre storage backend.

Développez pour afficher les étapes de configuration

1. Effectuez les opérations suivantes dans Trident si le volume ANF a été créé avant la mise à niveau vers Trident 24.10 :

   a. Activez le répertoire de snapshots pour chaque PV qui est 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 des instantanés 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 régulière, 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 à sauvegarder.

Résultat

L'application est prête pour la sauvegarde et la restauration à l'aide de Trident Protect. Chaque PVC est également disponible pour être utilisé par d'autres applications pour les sauvegardes et les restaurations.

Restaurer les 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

- Lors de la restauration d'une application, tous les points d'exécution configurés pour l'application sont restaurés avec l'application. Si un point 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. Cependant, la restauration à partir d'un instantané 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 les paramètres avancés pour personnaliser les opérations de restauration. Pour en savoir plus, consultez "Utilisez les paramètres de restauration avancés de Trident Protect".

Restaurer à partir d'une sauvegarde vers un espace de noms différent

Lorsque vous restaurez une sauvegarde dans un espace de noms différent à l'aide d'une BackupRestore CR, Trident Protect restaure l'application dans un nouvel espace de noms et crée un CR d'application pour l'application restaurée. Pour protéger l'application restaurée, créez des sauvegardes ou des instantanés à la demande, ou définissez une planification de protection.

REMARQUE

- La restauration d'une sauvegarde dans un espace de noms différent contenant des ressources existantes ne modifiera pas les ressources portant le même nom que celles 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.
- Lors de l'utilisation d'un CR pour restaurer dans un nouvel espace de noms, vous devez créer manuellement l'espace de noms de destination avant d'appliquer le CR. Trident Protect crée automatiquement les espaces de noms uniquement lors de l'utilisation du CLI.

Avant de commencer

Assurez-vous que la durée de validité du jeton de session AWS est suffisante pour toute opération de restauration s3 de longue durée. Si le jeton expire pendant l'opération de restauration, l'opération peut échouer.

- Consultez la "Documentation AWS API" pour plus d'informations sur la vérification de l'expiration du jeton de session actuel.
- Consultez la "Documentation AWS IAM" pour plus d'informations sur les identifiants relatifs aux ressources AWS.

REMARQUE

Lorsque vous restaurez des sauvegardes en utilisant Kopia comme moteur de déplacement de données, vous pouvez éventuellement spécifier des annotations dans le CR ou en utilisant la CLI pour contrôler le comportement du stockage temporaire utilisé par Kopia. Reportez-vous à l' "Documentation Kopia" pour plus d'informations sur les options que vous pouvez configurer. Utilisez la commande `tridentctl-protect create --help` pour plus d'informations sur la spécification des annotations avec la CLI Trident Protect.

Utilisez un 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**: (*Obligatoire*) Le nom de cette ressource personnalisée; choisissez un nom unique et pertinent pour votre environnement.
   - **spec.appArchivePath** : Le chemin à l'intérieur de AppVault où le contenu de la sauvegarde est stocké. 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** : (*Obligatoire*) Le nom du AppVault où les contenus de sauvegarde sont stockés.
   - **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** : La correspondance de l'espace de noms source de l'opération de restauration avec l'espace de noms de destination. Remplacez `my-source-namespace` et `my-destination-namespace` par les informations 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 fonction de leur relation avec les ressources que vous sélectionnez. Par exemple, si vous sélectionnez une ressource de type revendication de volume persistant et qu'elle possède un pod associé, Trident Protect restaurera également le pod associé.

- **resourceFilter.resourceSelectionCriteria** : (Obligatoire pour le filtrage) Utilisez `Include` ou `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 selon une opération OU, et les champs à l'intérieur de chaque élément (group, kind, version) correspondent selon une opération ET.
    - **resourceMatchers[].group**: (*Optionnel*) Groupe de la ressource à filtrer.
    - **resourceMatchers[].kind**: (*Optionnel*) Type de ressource à filtrer.
    - **resourceMatchers[].version**: (*Optionnel*) Version de la ressource à filtrer.
    - **resourceMatchers[].names**: (*Optionnel*) Noms dans le champ Kubernetes metadata.name de la ressource à filtrer.
    - **resourceMatchers[].namespaces**: (*Optionnel*) Espaces de noms dans le champ metadata.name de Kubernetes 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"]
```

4. Après avoir rempli le fichier `trident-protect-backup-restore-cr.yaml` avec les valeurs correctes, appliquez le CR :

```
kubectl apply -f trident-protect-backup-restore-cr.yaml
```

Utilisez la ligne de commandes (CLI)

Étapes

1. Restaurez la sauvegarde dans un espace de noms différent, en remplaçant les valeurs entre crochets par les informations de votre environnement. L' `namespace-mapping` argument utilise des espaces de noms séparés par des deux-points pour faire correspondre les espaces de noms source aux espaces de noms de destination corrects au 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 la durée de validité du jeton de session AWS est suffisante pour toute opération de restauration s3 de longue durée. Si le jeton expire pendant l'opération de restauration, l'opération peut échouer.

- Consultez la "Documentation AWS API" pour plus d'informations sur la vérification de l'expiration du jeton de session actuel.
- Consultez la "Documentation AWS IAM" pour plus d'informations sur les identifiants relatifs aux ressources AWS.

REMARQUE

Lorsque vous restaurez des sauvegardes en utilisant Kopia comme moteur de déplacement de données, vous pouvez éventuellement spécifier des annotations dans le CR ou en utilisant la CLI pour contrôler le comportement du stockage temporaire utilisé par Kopia. Reportez-vous à l' "Documentation Kopia" pour plus d'informations sur les options que vous pouvez configurer. Utilisez la commande `tridentctl-protect create --help` pour plus d'informations sur la spécification des annotations avec la CLI Trident Protect.

Utilisez un 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**: (*Obligatoire*) Le nom de cette ressource personnalisée; choisissez un nom unique et pertinent pour votre environnement.
   - **spec.appArchivePath** : Le chemin à l'intérieur de AppVault où le contenu de la sauvegarde est stocké. 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** : (*Obligatoire*) Le nom du AppVault où les contenus de sauvegarde sont stockés.

     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 fonction de leur relation avec les ressources que vous sélectionnez. Par exemple, si vous sélectionnez une ressource de type revendication de volume persistant et qu'elle possède un pod associé, Trident Protect restaurera également le pod associé.

   - **resourceFilter.resourceSelectionCriteria** : (Obligatoire pour le filtrage) Utilisez `Include` ou `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 selon une opération OU, et les champs à l'intérieur de chaque élément (group, kind, version) correspondent selon une opération ET.
       - **resourceMatchers[].group**: (*Optionnel*) Groupe de la ressource à filtrer.

- **resourceMatchers[].kind**: (*Optionnel*) Type de ressource à filtrer.
- **resourceMatchers[].version**: (*Optionnel*) Version de la ressource à filtrer.
- **resourceMatchers[].names**: (*Optionnel*) Noms dans le champ Kubernetes metadata.name de la ressource à filtrer.
- **resourceMatchers[].namespaces**: (*Optionnel*) Espaces de noms dans le champ metadata.name de Kubernetes 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"]
```

4. Après avoir rempli le fichier `trident-protect-backup-ipr-cr.yaml` avec les valeurs correctes, appliquez le CR :

```
kubectl apply -f trident-protect-backup-ipr-cr.yaml
```

Utilisez la ligne de commandes (CLI)

Étapes

1. Restaurez la sauvegarde dans l'espace de noms d'origine, en remplaçant les valeurs entre crochets par les informations de votre environnement. L'`backup`argument utilise un 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>
```

Restaurer à partir d'une sauvegarde vers un cluster différent

Vous pouvez restaurer une sauvegarde sur un cluster différent en cas de problème avec le cluster d'origine.

REMARQUE

- Lorsque vous restaurez des sauvegardes en utilisant Kopia comme moteur de déplacement de données, vous pouvez éventuellement spécifier des annotations dans le CR ou en utilisant la CLI pour contrôler le comportement du stockage temporaire utilisé par Kopia. Reportez-vous à l' "Documentation Kopia" pour plus d'informations sur les options que vous pouvez configurer. Utilisez la commande `tridentctl-protect create --help` pour plus d'informations sur la spécification des annotations avec la CLI Trident Protect.
- Lors de l'utilisation d'un CR pour restaurer dans un nouvel espace de noms, vous devez créer manuellement l'espace de noms de destination avant d'appliquer le CR. Trident Protect crée automatiquement les espaces de noms uniquement lors de l'utilisation du CLI.

Avant de commencer

Assurez-vous que les conditions préalables suivantes sont remplies :

- Le cluster de destination a Trident Protect installé.
- Le cluster de destination a accès au chemin du 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 le CR AppVault lors de l'exécution de la commande `tridentctl-protect get appvaultcontent`. Si des restrictions réseau empêchent l'accès, exécutez la CLI Trident Protect depuis un pod sur le cluster de destination à la place.
- Assurez-vous que la durée de validité du jeton de session AWS soit suffisante pour toute opération de restauration de longue durée. Si le jeton expire pendant l'opération de restauration, l'opération peut échouer.
  - Consultez la "Documentation AWS API" pour plus d'informations sur la vérification de l'expiration du jeton de session actuel.
  - Consultez la "Documentation AWS" pour plus d'informations sur les identifiants relatifs aux ressources 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 des applications correspondantes, les horodatages et les chemins d'accès aux archives.

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 en utilisant le nom AppVault et le chemin d'accès à l'archive :

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.

Utilisez un 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**: (*Obligatoire*) Le nom de cette ressource personnalisée; choisissez un nom unique et pertinent pour votre environnement.
   - **spec.appVaultRef** : (*Obligatoire*) Le nom du AppVault où les contenus de sauvegarde sont stockés.
   - **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** : La correspondance de l'espace de noms source de l'opération de restauration avec l'espace de noms de destination. Remplacez `my-source-namespace` et `my-destination-namespace` par les informations 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. Après avoir rempli le fichier `trident-protect-backup-restore-cr.yaml` avec les valeurs correctes, appliquez le CR :

```
kubectl apply -f trident-protect-backup-restore-cr.yaml
```

Utilisez la ligne de commandes (CLI)

1. Utilisez la commande suivante pour restaurer l'application, en remplaçant les valeurs entre crochets par les informations de votre environnement. L'argument namespace-mapping utilise des espaces de noms séparés par des deux-points pour associer les espaces de noms source aux espaces de noms de destination correspondants, 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>
```

Restaurer à partir d'un instantané vers un espace de noms différent

Vous pouvez restaurer des données à partir d'un instantané à l'aide d'un fichier de ressource personnalisé (CR), soit vers un autre espace de noms, soit vers l'espace de noms source d'origine. Lorsque vous restaurez un instantané vers un autre espace de noms à l'aide d'un SnapshotRestore CR, Trident Protect restaure l'application dans un nouvel espace de noms et crée un CR d'application pour l'application restaurée. Pour protéger l'application restaurée, créez des sauvegardes ou des instantanés à la demande, ou définissez une planification de protection.

REMARQUE

- SnapshotRestore prend en charge l' `spec.storageClassMapping`attribut, mais uniquement lorsque les classes de stockage source et de destination utilisent le même système de stockage. Si vous tentez de restaurer vers une classe de stockage `StorageClass` qui utilise un système de stockage différent, l'opération de restauration échouera.
- Lors de l'utilisation d'un CR pour restaurer dans un nouvel espace de noms, vous devez créer manuellement l'espace de noms de destination avant d'appliquer le CR. Trident Protect crée automatiquement les espaces de noms uniquement lors de l'utilisation du CLI.

Avant de commencer

Assurez-vous que la durée de validité du jeton de session AWS est suffisante pour toute opération de restauration s3 de longue durée. Si le jeton expire pendant l'opération de restauration, l'opération peut échouer.

- Consultez la "Documentation AWS API" pour plus d'informations sur la vérification de l'expiration du jeton de session actuel.
- Consultez la "Documentation AWS IAM" pour plus d'informations sur les identifiants relatifs aux ressources AWS.

Utilisez un 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**: (*Obligatoire*) Le nom de cette ressource personnalisée; choisissez un nom unique et pertinent pour votre environnement.
   - **spec.appVaultRef** : (*Obligatoire*) Le nom du AppVault où le contenu de l'instantané est stocké.
   - **spec.appArchivePath** : Le chemin à l'intérieur de AppVault où les contenus de l'instantané sont stockés. 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** : La correspondance de l'espace de noms source de l'opération de restauration avec l'espace de noms de destination. Remplacez `my-source-namespace` et `my-destination-namespace` par les informations de votre environnement.

     ```
     ---
     apiVersion: protect.trident.netapp.io/v1
     kind: SnapshotRestore
     metadata:
       name: my-cr-name
       namespace: my-app-namespace
     spec:
       appVaultRef: appvault-name
       appArchivePath: my-snapshot-path
       namespaceMapping: [{"source": "my-source-namespace",
     "destination": "my-destination-namespace"}]
     ```

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 fonction de leur relation avec les ressources que vous sélectionnez. Par exemple, si vous sélectionnez une ressource de type revendication de volume persistant et qu'elle possède un pod associé, Trident Protect restaurera également le pod associé.

   - **resourceFilter.resourceSelectionCriteria** : (Obligatoire pour le filtrage) Utilisez `Include` ou `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 selon une opération OU, et les champs à l'intérieur de chaque élément (group, kind, version) correspondent selon une opération ET.
  - **resourceMatchers[].group**: (*Optionnel*) Groupe de la ressource à filtrer.
  - **resourceMatchers[].kind**: (*Optionnel*) Type de ressource à filtrer.
  - **resourceMatchers[].version**: (*Optionnel*) Version de la ressource à filtrer.
  - **resourceMatchers[].names**: (*Optionnel*) Noms dans le champ Kubernetes metadata.name de la ressource à filtrer.
  - **resourceMatchers[].namespaces**: (*Optionnel*) Espaces de noms dans le champ metadata.name de Kubernetes 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"]
```

4. Après avoir rempli le fichier `trident-protect-snapshot-restore-cr.yaml` avec les valeurs correctes, appliquez le CR :

```
kubectl apply -f trident-protect-snapshot-restore-cr.yaml
```

Utilisez la ligne de commandes (CLI)

Étapes

1. Restaurez l'instantané dans un espace de noms différent, en remplaçant les valeurs entre crochets par les informations de votre environnement.

- L'`snapshot`argument utilise un espace de noms et un nom d'instantané au format `<namespace>/<name>.
- L'`namespace-mapping`argument utilise des espaces de noms séparés par des deux-points pour faire correspondre les espaces de noms source aux espaces de noms de destination corrects au 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 instantané 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 la durée de validité du jeton de session AWS est suffisante pour toute opération de restauration s3 de longue durée. Si le jeton expire pendant l'opération de restauration, l'opération peut échouer.

- Consultez la "Documentation AWS API" pour plus d'informations sur la vérification de l'expiration du jeton de session actuel.
- Consultez la "Documentation AWS IAM" pour plus d'informations sur les identifiants relatifs aux ressources AWS.

Utilisez un 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**: (*Obligatoire*) Le nom de cette ressource personnalisée; choisissez un nom unique et pertinent pour votre environnement.
   - **spec.appVaultRef** : (*Obligatoire*) Le nom du AppVault où le contenu de l'instantané est stocké.
   - **spec.appArchivePath** : Le chemin à l'intérieur de AppVault où les contenus de l'instantané sont stockés. Vous pouvez utiliser la commande suivante pour trouver ce chemin :

     ```
     kubectl get snapshots <SNAPSHOT_NAME> -n my-app-namespace -o jsonpath='{.status.appArchivePath}'
     ```

     ```
     ---
     apiVersion: protect.trident.netapp.io/v1
     kind: SnapshotInplaceRestore
     metadata:
       name: my-cr-name
       namespace: my-app-namespace
     spec:
       appVaultRef: appvault-name
       appArchivePath: my-snapshot-path
     ```

3. (*Facultatif*) Si vous devez sélectionner uniquement certaines ressources de l'application à restaurer, ajoutez un filtrage qui inclut ou exclut les ressources marquées avec des étiquettes particulières :

   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 type revendication de volume persistant et qu'elle possède un pod associé, Trident Protect restaurera également le pod associé.

   - **resourceFilter.resourceSelectionCriteria** : (Obligatoire pour le filtrage) Utilisez `Include` ou `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 selon une opération OU, et les champs à l'intérieur de chaque élément (group, kind, version) correspondent selon une opération ET.
       - **resourceMatchers[].group**: (*Optionnel*) Groupe de la ressource à filtrer.
       - **resourceMatchers[].kind**: (*Optionnel*) Type de ressource à filtrer.
       - **resourceMatchers[].version**: (*Optionnel*) Version de la ressource à filtrer.

- **resourceMatchers[].names**: (*Optionnel*) Noms dans le champ Kubernetes metadata.name de la ressource à filtrer.
- **resourceMatchers[].namespaces**: (*Optionnel*) Espaces de noms dans le champ metadata.name de Kubernetes 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"]
```

4. Après avoir rempli le fichier `trident-protect-snapshot-ipr-cr.yaml` avec les valeurs correctes, appliquez le CR :

```
kubectl apply -f trident-protect-snapshot-ipr-cr.yaml
```

Utilisez la ligne de commandes (CLI)

Étapes

1. Restaurez l'instantané dans l'espace de noms d'origine, en remplaçant les valeurs entre crochets 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 commandes pour vérifier l'état d'une opération de restauration qui est en cours, terminée ou ayant échoué.

Étapes

1. Utilisez la commande suivante pour récupérer l'état de l'opération de restauration, en remplaçant les valeurs entre crochets par les 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 de namespace et les options de stockage pour répondre à vos besoins spécifiques.

Annotations et étiquettes d'espace de noms lors des opérations de restauration et de basculement

Lors des opérations de restauration et de basculement, les étiquettes et annotations de l'espace de noms de destination sont mises à jour pour correspondre aux étiquettes et annotations de l'espace de noms source. Les étiquettes ou annotations 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 déjà présentes sont remplacées pour correspondre à la valeur de l'espace de noms source. Les étiquettes ou annotations qui existent uniquement dans l'espace de noms de destination restent inchangées.

REMARQUE	Si vous utilisez Red Hat OpenShift, il est important de noter le rôle crucial des annotations d'espace de noms dans les environnements OpenShift. Les annotations d'espace de noms garantissent que les pods restaurés respectent les permissions et les configurations de sécurité appropriées définies par les contraintes de contexte de sécurité (OpenShift SCC) et peuvent accéder aux volumes sans problème de permissions. Pour plus d'informations, consultez la "Documentation des contraintes de contexte de sécurité OpenShift".

Vous pouvez empêcher l'écrasement de certaines annotations dans l'espace de noms de destination en définissant 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, 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 des paramètres supplémentaires du chart Helm Trident Protect".

Si vous avez installé l'application source avec Helm avec le `--create-namespace` flag, un traitement spécial est appliqué à la clé de label `name`. Lors du processus de restauration ou de basculement, Trident Protect copie ce label dans l'espace de noms de destination, mais met à jour la valeur avec celle de l'espace de noms de destination si la valeur provenant 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 un espace de noms de destination, chacun possédant des annotations et des étiquettes différentes. Vous pouvez voir l'état de l'espace de noms de destination avant et après l'opération, et comment 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 des espaces de noms source et de destination de l'exemple avant l'opération de restauration ou de basculement :

Espace de noms	Annotations	Étiquettes
Espace de noms ns-1 (source)	• annotation.one/key: "updatedvalue" • annotation.two/key: "true"	• environment=production • conformité=hipaa • name=ns-1
Espace de noms ns-2 (destination)	• annotation.one/key: "true" • annotation.three/key: "false"	• rôle=base de données

Après l'opération de restauration

Le tableau suivant illustre l'état de l'espace de noms de destination après l'opération de restauration ou de basculement. Certaines clés ont été ajoutées, d'autres ont été écrasées, et l' `name`étiquette a été mise 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.two/key: "true" • annotation.three/key: "false"	• name=ns-2 • conformité=hipaa • environment=production • rôle=base de données

Champs pris en charge

Cette section décrit les champs supplémentaires disponibles pour les opérations de restauration.

Correspondance des classes de stockage

L' `spec.storageClassMapping`attribut définit une correspondance entre une classe de stockage présente dans l'application source et une nouvelle classe de stockage sur le cluster cible. Vous pouvez l'utiliser lors de la migration d'applications entre des clusters avec des classes de stockage différentes ou lors du changement de système 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 l'opération de déplacement de données soit bloquée.	"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	Temps maximal (en secondes) d'attente pour que tout nouveau PersistentVolumeClaims (PVC) atteigne la phase `Bound` avant que l'opération n'échoue. S'applique à tous les types de CR de restauration (BackupRestore, BackupInplaceRestore, SnapshotRestore, SnapshotInplaceRestore). Utilisez une valeur plus élevée si votre backend de stockage ou votre cluster nécessite souvent plus de temps.	"1200" (20 minutes)

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 d'espace de noms lors des opérations de restauration et de basculement

Lors des opérations de restauration et de basculement, les étiquettes et annotations de l'espace de noms de destination sont mises à jour pour correspondre aux étiquettes et annotations de l'espace de noms source. Les étiquettes ou annotations 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 déjà présentes sont remplacées pour correspondre à la valeur de l'espace de noms source. Les étiquettes ou annotations qui existent uniquement dans l'espace de noms de destination restent inchangées.

REMARQUE Si vous utilisez Red Hat OpenShift, il est important de noter le rôle crucial des annotations d'espace de noms dans les environnements OpenShift. Les annotations d'espace de noms garantissent que les pods restaurés respectent les permissions et les configurations de sécurité appropriées définies par les contraintes de contexte de sécurité (OpenShift SCC) et peuvent accéder aux volumes sans problème de permissions. Pour plus d'informations, consultez la "Documentation des contraintes de contexte de sécurité OpenShift".

Vous pouvez empêcher l'écrasement de certaines annotations dans l'espace de noms de destination en définissant 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, 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 des paramètres supplémentaires du chart Helm Trident Protect".

Si vous avez installé l'application source avec Helm avec le `--create-namespace` flag, un traitement spécial est appliqué à la clé de label `name`. Lors du processus de restauration ou de basculement, Trident Protect copie ce label dans l'espace de noms de destination, mais met à jour la valeur avec celle de l'espace de noms de destination si la valeur provenant 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 un espace de noms de destination, chacun possédant des annotations et des étiquettes différentes. Vous pouvez voir l'état de l'espace de noms de destination avant et après l'opération, et comment 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 des espaces de noms source et de destination de l'exemple avant l'opération de restauration ou de basculement :

Espace de noms	Annotations	Étiquettes
Espace de noms ns-1 (source)	• annotation.one/key: "updatedvalue" • annotation.two/key: "true"	• environment=production • conformité=hipaa • name=ns-1
Espace de noms ns-2 (destination)	• annotation.one/key: "true" • annotation.three/key: "false"	• rôle=base de données

Après l'opération de restauration

Le tableau suivant illustre l'état de l'espace de noms de destination après l'opération de restauration ou de basculement. Certaines clés ont été ajoutées, d'autres ont été écrasées, et l' `name`étiquette a été mise 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.two/key: "true" • annotation.three/key: "false"	• name=ns-2 • conformité=hipaa • environment=production • rôle=base de données

REMARQUE Vous pouvez configurer Trident Protect pour geler et dégeler les systèmes de fichiers pendant les opérations de protection des données. "En savoir plus sur la configuration du gel du système de fichiers avec Trident Protect".

Hooks d'exécution lors des opérations de basculement et d'inversion

Lorsque vous utilisez une 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 d'un 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 sur l'application et s'exécuteront lors de toute action concernée.
- Lors d'une opération inverse ou d'une resynchronisation inverse, 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 afin d'empêcher leur exécution.

Pour en savoir plus sur les hooks d'exécution, reportez-vous à "Gérer les hooks d'exécution de Trident Protect".

Configurer une relation de réplication

La mise en place d'une relation de réplication implique les éléments suivants :

- Choisir la fréquence à laquelle vous souhaitez que Trident Protect prenne 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 du calendrier de réplication (inclut les ressources Kubernetes ainsi que les données de volume persistant)
- Définition de l'heure à laquelle le snapshot sera pris

Étapes

1. Sur le cluster source, créez un AppVault pour l'application source. En fonction de votre fournisseur de stockage, modifiez un exemple dans "Ressources personnalisées AppVault" pour l'adapter à votre environnement :

Créez un AppVault en utilisant un 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** : (*Obligatoire*) Le nom de la ressource personnalisée AppVault. Notez le nom que vous choisissez, car d'autres fichiers CR nécessaires pour une relation de réplication font référence à cette valeur.
- **spec.providerConfig** : (*Obligatoire*) Stocke la configuration nécessaire pour accéder à AppVault en utilisant le fournisseur spécifié. Choisissez un bucketName et tout autre détail nécessaire pour votre fournisseur. Notez les valeurs que vous choisissez, car d'autres fichiers CR nécessaires pour une relation de réplication font référence à ces valeurs. Reportez-vous à "Ressources personnalisées AppVault" pour des exemples de CR AppVault avec d'autres fournisseurs.
- **spec.providerCredentials** : (*Obligatoire*) Stocke les références à tout identifiant requis pour accéder à AppVault en utilisant le fournisseur spécifié.
  - **spec.providerCredentials.valueFromSecret** : (*Obligatoire*) Indique que la valeur de l'identifiant doit provenir d'un secret.
    - **clé**: (*Obligatoire*) La clé valide du secret à sélectionner.
    - **nom** : (*Obligatoire*) Nom du secret contenant la valeur de ce champ. Doit se trouver dans le même espace de noms.
  - **spec.providerCredentials.secretAccessKey** : (*Obligatoire*) La clé d'accès utilisée pour accéder au fournisseur. Le **nom** doit correspondre à **spec.providerCredentials.valueFromSecret.name**.
- **spec.providerType** : (Obligatoire) Détermine ce qui fournit la sauvegarde ; par exemple, NetApp ONTAP S3, S3 générique, Google Cloud ou Microsoft Azure. Valeurs possibles :
  - aws
  - azure
  - gcp
  - generic-s3
  - ontap-s3
  - storagegrid-s3

c. Après avoir rempli le fichier `trident-protect-appvault-primary-source.yaml` avec les valeurs correctes, appliquez le CR :

```
kubectl apply -f trident-protect-appvault-primary-source.yaml -n
trident-protect
```

Créez un AppVault à l'aide de la CLI

a. Créez le AppVault, en remplaçant les valeurs entre crochets par les informations provenant 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 le CR d'application source :

Créez l'application source à l'aide d'une CR

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** : (*Obligatoire*) Le nom de la ressource personnalisée de l'application. Notez le nom que vous choisissez, car d'autres fichiers CR nécessaires pour une relation de réplication font référence à cette valeur.
- **spec.includedNamespaces** : (*Obligatoire*) Un tableau des espaces de noms et des étiquettes associées. Utilisez les noms des espaces de noms et, éventuellement, restreignez la portée des espaces de noms avec des étiquettes pour spécifier les ressources qui existent dans les espaces de noms listés ici. L'espace de noms de l'application doit faire partie de ce tableau.

  Exemple de 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. Après avoir rempli le fichier `trident-protect-app-source.yaml` avec les valeurs correctes, appliquez le CR :

```
kubectl apply -f trident-protect-app-source.yaml -n my-app-namespace
```

Créez l'application source à l'aide de la CLI

a. Créez l'application source. Par exemple :

```
tridentctl-protect create app <my-app-name> --namespaces <namespaces-to-be-included> -n <my-app-namespace>
```

3. Optionnellement, sur le cluster source, prenez un instantané de l'application source. Cet instantané est utilisé comme base pour l'application sur le cluster de destination. Si vous ignorez cette étape, vous devrez attendre la prochaine exécution de l'instantané planifié afin d'avoir un instantané récent. Pour créer un instantané à la demande, consultez "Créer un instantané à la demande".

4. Sur le cluster source, créez la CR de planification de réplication :

En complément du calendrier ci-dessous, il est recommandé de créer un calendrier de snapshot quotidien distinct, avec une période de conservation de 7 jours, afin de maintenir un snapshot commun entre les clusters ONTAP appariés. Cela garantit que les snapshots sont disponibles pendant jusqu'à 7 jours, mais la période de conservation peut être personnalisée en fonction des besoins de l'utilisateur.

REMARQUE

En cas de basculement, le système peut utiliser ces instantanés pendant 7 jours maximum pour les opérations de restauration. Cette approche rend le processus de restauration plus rapide et plus efficace, car seules les modifications apportées depuis le dernier instantané seront transférées, pas toutes les 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'un CR

a. Créez une planification 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**: (*Obligatoire*) Le nom de la ressource personnalisée de planification.
- **spéc.appVaultRef** : (*Obligatoire*) Cette valeur doit correspondre au champ metadata.name de AppVault pour l'application source.
- **spec.applicationRef**: (*Obligatoire*) Cette valeur doit correspondre au champ metadata.name de l'application source CR.
- **spec.backupRetention** : (*Obligatoire*) Ce champ est obligatoire et la valeur doit être définie sur 0.
- **spec.enabled** : doit être défini sur true.
- **spec.granularité** : Doit être définie sur `Custom`.
- **spec.recurrenceRule** : Définir une date de début en temps UTC et un intervalle de récurrence.
- **spec.snapshotRetention** : Doit être réglé 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. Après avoir rempli le fichier `trident-protect-schedule.yaml` avec les valeurs correctes, appliquez le 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 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 AppVault CR d'application source identique à la AppVault CR que vous avez appliquée sur le cluster source et nommez-la (par exemple, `trident-protect-appvault-primary-destination.yaml`).
6. Appliquez le CR :

```
kubectl apply -f trident-protect-appvault-primary-destination.yaml -n
trident-protect
```

7. Créez une ressource personnalisée AppVault CR de destination pour l'application de destination sur le cluster de destination. En fonction de votre fournisseur de stockage, modifiez un exemple dans "Ressources personnalisées AppVault" pour l'adapter à 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** : (*Obligatoire*) Le nom de la ressource personnalisée AppVault. Notez le nom que vous choisissez, car d'autres fichiers CR nécessaires pour une relation de réplication font référence à cette valeur.
      - **spec.providerConfig** : (*Obligatoire*) Stocke la configuration nécessaire pour accéder à AppVault en utilisant le fournisseur spécifié. Choisissez un `bucketName` et tous les autres détails nécessaires pour votre fournisseur. Notez les valeurs que vous choisissez, car d'autres fichiers CR nécessaires pour une relation de réplication font référence à ces valeurs. Reportez-vous à

"Ressources personnalisées AppVault" pour des exemples de CR AppVault avec d'autres fournisseurs.

- **spec.providerCredentials** : (*Obligatoire*) Stocke les références à tout identifiant requis pour accéder à AppVault en utilisant le fournisseur spécifié.
  - **spec.providerCredentials.valueFromSecret** : (*Obligatoire*) Indique que la valeur de l'identifiant doit provenir d'un secret.
    - **clé**: (*Obligatoire*) La clé valide du secret à sélectionner.
    - **nom** : (*Obligatoire*) Nom du secret contenant la valeur de ce champ. Doit se trouver dans le même espace de noms.
  - **spec.providerCredentials.secretAccessKey** : (*Obligatoire*) La clé d'accès utilisée pour accéder au fournisseur. Le **nom** doit correspondre à **spec.providerCredentials.valueFromSecret.name**.
- **spec.providerType** : (Obligatoire) Détermine ce qui fournit la sauvegarde ; par exemple, NetApp ONTAP S3, S3 générique, Google Cloud ou Microsoft Azure. Valeurs possibles :
  - aws
  - azure
  - gcp
  - generic-s3
  - ontap-s3
  - storagegrid-s3

c. Après avoir rempli le fichier `trident-protect-appvault-secondary-destination.yaml` avec les valeurs correctes, appliquez le 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'un CR, créez manuellement l'espace de noms de destination avant d'appliquer le CR. Trident Protect crée automatiquement les espaces de noms uniquement lors de l'utilisation du CLI.

Créez un AppMirrorRelationship à l'aide d'un 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**: (*Obligatoire*) Cette valeur doit correspondre au nom de AppVault pour l'application de destination sur le cluster de destination.
- **spec.namespaceMapping**: (*Obligatoire*) Les espaces de noms de destination et de source doivent correspondre à l'espace de noms de l'application défini dans la CR de l'application respective.
- **spec.sourceAppVaultRef**: (*Obligatoire*) Cette valeur doit correspondre au nom de AppVault pour l'application source.
- **spec.sourceApplicationName**: (*Obligatoire*) 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** : (*Facultatif*) 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 appairée avec l'environnement source. Si la classe de stockage n'est pas fournie, la classe de stockage par défaut du cluster sera utilisée par défaut.
- **spec.recurrenceRule** : Définir une date de début en temps 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. Après avoir rempli le fichier `trident-protect-relationship.yaml` avec les valeurs correctes, appliquez le CR :

```
kubectl apply -f trident-protect-relationship.yaml -n my-app-
namespace
```

Créez un AppMirrorRelationship à l'aide de la CLI

a. Créez et appliquez l'objet AppMirrorRelationship, en remplaçant les valeurs entre crochets par les informations 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> -n <application_namespace>
```

Exemple :

```
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. (*Facultatif*) Sur le cluster de destination, vérifiez l'état et le statut 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 ne met pas l'application hors service 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 changez la valeur de **spec.desiredState** en `Promoted`.
2. Enregistrez le fichier CR.
3. Appliquez le CR :

```
kubectl apply -f trident-protect-relationship.yaml -n my-app-namespace
```

4. (*Facultatif*) Créez tous les plans de protection dont vous avez besoin sur l'application basculée.
5. (*Facultatif*) Vérifiez l'état et le statut de la relation de réplication :

```
kubectl get amr -n my-app-namespace <relationship name> -o=jsonpath
='{.status}' | jq
```

Resynchroniser une relation de réplication basculée

L'opération de resynchronisation rétablit la relation de réplication. Après avoir effectué une opération de resynchronisation, 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 annulées.

Le processus arrête l'application sur le cluster de destination avant de rétablir la réplication.

IMPORTANT	Toutes les données écrites dans l'application de destination pendant le basculement seront perdues.

Étapes

1. Facultatif : sur le cluster source, créez un instantané de l'application source. Cela garantit que les dernières modifications du cluster source sont prises en compte.
2. Sur le cluster de destination, modifiez le fichier CR AppMirrorRelationship (par exemple, `trident-protect-relationship.yaml`) et changez la valeur de spec.desiredState à `Established`.
3. Enregistrez le fichier CR.
4. Appliquez le CR :

```
kubectl apply -f trident-protect-relationship.yaml -n my-app-namespace
```

5. Si vous avez créé des planifications de protection sur le cluster de destination pour protéger l'application basculée, supprimez-les. Toute planification restante entraîne des échecs de snapshots de volume.

Resynchroniser à l'inverse une relation de réplication ayant échoué

Lorsque vous effectuez une resynchronisation inverse d'une relation de réplication ayant basculé, 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 le CR AppMirrorRelationship. Cela fait en sorte que la destination devienne la source. S'il reste des planifications de protection sur le nouveau cluster de destination, supprimez-les.
2. Établissez une relation de réplication en appliquant les fichiers CR que vous avez initialement utilisés pour configurer la relation aux clusters opposés.
3. Assurez-vous que la nouvelle destination (cluster source d'origine) est configurée avec les deux AppVault CR.
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 cette situation, vous échangez la source et la destination.

Étapes

1. Sur le cluster source, créez un instantané d'arrêt :

Créez un instantané d'arrêt à l'aide d'une CR

a. Désactivez les plannings de la stratégie de protection pour l'application source.

b. Créez un fichier CR ShutdownSnapshot :

   i. Créez le fichier de ressource personnalisée (CR) et nommez-le (par exemple, `trident-protect-shutdownsnapshot.yaml`).

   ii. Configurez les attributs suivants :

   - **metadata.name**: (*Obligatoire*) Le nom de la ressource personnalisée.
   - **spec.AppVaultRef** : (*Obligatoire*) Cette valeur doit correspondre au champ metadata.name de AppVault pour l'application source.
   - **spec.ApplicationRef** : (*Obligatoire*) 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. Après avoir rempli le fichier `trident-protect-shutdownsnapshot.yaml` avec les valeurs correctes, appliquez le CR :

```
kubectl apply -f trident-protect-shutdownsnapshot.yaml -n my-app-namespace
```

Créez un instantané d'arrêt à l'aide de la CLI

a. Créez l'instantané d'arrêt, en remplaçant les valeurs entre crochets 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, après la fin de la capture instantanée de l'arrêt, obtenez l'état de la capture instantanée de l'arrêt :

```
kubectl get shutdownsnapshot -n my-app-namespace
<shutdown_snapshot_name> -o yaml
```

3. Sur le cluster source, trouvez la valeur de **shutdownsnapshot.status.appArchivePath** à l'aide de la commande suivante, et notez la dernière partie du chemin d'accès au fichier (également appelée le basename ; cela correspond à tout ce qui suit 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 champ `spec.promotedSnapshot` 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 inverses dans Resynchroniser à l'inverse 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 capture instantanée est prise des ressources Kubernetes de l'application source d'origine.
- Les pods de l'application source d'origine sont arrêtés en douceur en supprimant les ressources Kubernetes de l'application (en laissant les PVC et les PV en place).
- Une fois les pods arrêtés, des instantanés des volumes de l'app sont pris et répliqués.
- Les relations SnapMirror sont rompues, ce qui rend les volumes de destination prêts pour la lecture/écriture.
- Les ressources Kubernetes de l'application sont restaurées à partir de l'instantané pris avant l'arrêt, en utilisant les données de volume répliquées après l'arrêt de l'application source d'origine.
- La réplication est rétablie dans le sens inverse.

Rétablir les applications sur le cluster source d'origine

Avec Trident Protect, vous pouvez effectuer un « retour arrière » après une opération de basculement en utilisant la séquence d'opérations suivante. Dans ce workflow pour restaurer la direction de réplication d'origine, Trident Protect réplique (resynchronise) toutes les modifications de l'application vers l'application source d'origine avant d'inverser la direction de la réplication.

Ce processus débute à partir d'une relation ayant effectué un basculement vers une destination et comprend les étapes suivantes :

- Commencez par un état de basculement échoué.
- Inversez la resynchronisation de la relation de réplication.

AVERTISSEMENT	N'effectuez pas d'opération de resynchronisation normale, car cela supprimerait les données écrites sur le cluster de destination pendant la procédure de basculement.

- Inversez le sens de la réplication.

Étapes

1. Effectuez les Resynchroniser à l'inverse une relation de réplication ayant échoué étapes.
2. Effectuez 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'une application, il en résulte deux applications distinctes sans aucune relation entre elles.

Étapes

1. Sur le cluster de destination actuel, supprimez le 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 des données de sauvegarde.

REMARQUE	Lors de la migration d'une application, tous les points d'exécution configurés pour l'application sont migrés avec l'application. Si un point 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 pour les scénarios suivants, vous pouvez automatiser des tâches spécifiques de sauvegarde et de restauration.

Cloner vers le même cluster

Pour cloner une application sur le même cluster, créez un instantané ou une sauvegarde et restaurez les données sur le même cluster.

Étapes

1. Effectuez l'une des opérations suivantes :
   a. "Créer un instantané".
   b. "Créer une sauvegarde".
2. Sur le même cluster, effectuez l'une des opérations suivantes, selon que vous ayez créé un instantané ou une sauvegarde :

a. "Restaurez vos données à partir de l'instantané".

b. "Restaurez vos données à partir de la sauvegarde".

Cloner vers un cluster différent

Pour cloner une application sur un autre cluster (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 en utilisant "Réplication SnapMirror".

Étapes

1. "Créer une sauvegarde".
2. Assurez-vous que la AppVault CR du compartiment de stockage d'objets 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".

Migrer des applications d'une classe de stockage à une autre classe de stockage

Vous pouvez migrer des applications d'une classe de stockage à une autre en restaurant une sauvegarde dans la classe de stockage de destination.

Par exemple (en excluant les secrets du 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**: (*Obligatoire*) Le nom de cette ressource personnalisée; choisissez un nom unique et pertinent pour votre environnement.
   - **spec.appArchivePath** : Le chemin à l'intérieur de AppVault où les contenus de l'instantané sont stockés. 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** : (*Obligatoire*) Le nom du AppVault où le contenu de l'instantané est stocké.
   - **spec.namespaceMapping** : La correspondance de l'espace de noms source de l'opération de restauration avec l'espace de noms de destination. Remplacez `my-source-namespace` et `my-destination-namespace` par les informations 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. Optionnellement, si vous devez sélectionner uniquement certaines ressources de l'application à restaurer, ajoutez un filtrage qui inclut ou exclut les ressources marquées avec des étiquettes particulières :
   - **resourceFilter.resourceSelectionCriteria** : (Obligatoire 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 selon une opération OU, et les champs à l'intérieur de chaque élément (group, kind, version) correspondent selon une opération ET.
       - **resourceMatchers[].group**: (*Optionnel*) Groupe de la ressource à filtrer.
       - **resourceMatchers[].kind**: (*Optionnel*) Type de ressource à filtrer.
       - **resourceMatchers[].version**: (*Optionnel*) Version de la ressource à filtrer.

- **resourceMatchers[].names**: (*Optionnel*) Noms dans le champ Kubernetes metadata.name de la ressource à filtrer.
- **resourceMatchers[].namespaces**: (*Optionnel*) Espaces de noms dans le champ metadata.name de Kubernetes 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"]
```

4. Après avoir rempli le fichier `trident-protect-snapshot-restore-cr.yaml` avec les valeurs correctes, appliquez le CR :

```
kubectl apply -f trident-protect-snapshot-restore-cr.yaml
```

Restaurez l'instantané à l'aide de l'interface de ligne de commande (CLI)

Étapes

1. Restaurez l'instantané dans un espace de noms différent, en remplaçant les valeurs entre crochets par les informations de votre environnement.

   - L'`snapshot`argument utilise un espace de noms et un nom d'instantané au format `<namespace>/<name>.
   - L'`namespace-mapping`argument utilise des espaces de noms séparés par des deux-points pour faire correspondre les espaces de noms source aux espaces de noms de destination corrects au format `source1:dest1,source2:dest2.

   Par exemple :

```
tridentctl-protect create snapshotrestore <my_restore_name>
--snapshot <namespace/snapshot_to_restore> --namespace-mapping
<source_to_destination_namespace_mapping>
```

Gérer les hooks d'exécution de Trident Protect

Un point d'exécution est une action personnalisée que vous pouvez configurer pour qu'elle s'exécute conjointement avec une opération de protection des données d'une application gérée. Par exemple, si vous avez une application de base de données, vous pouvez utiliser un point d'exécution pour suspendre toutes les transactions de la base de données avant un instantané, puis reprendre les transactions après la fin de l'instantané. Cela garantit des instantanés cohérents au niveau des applications.

Types de hooks 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
- Après l'instantané
- Pré-sauvegarde
- Post-sauvegarde
- Après restauration
- Après le basculement

Ordre d'exécution

Lorsqu'une opération de protection des données est exécutée, les événements du hook d'exécution se produisent dans l'ordre suivant :

1. Les hooks d'exécution personnalisés applicables avant l'opération sont exécutés sur les conteneurs appropriés. Vous pouvez créer et exécuter autant de hooks d'exécution personnalisés que nécessaire, mais l'ordre d'exécution de ces hooks avant l'opération n'est ni garanti ni configurable.
2. Des blocages du système de fichiers peuvent survenir, le cas échéant. "En savoir plus 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 sont dégelés, le cas échéant.
5. Les hooks d'exécution post-opération personnalisés applicables sont exécutés sur les conteneurs appropriés. Vous pouvez créer et exécuter autant de hooks post-opération personnalisés que nécessaire, mais l'ordre d'exécution de ces hooks après l'opération n'est ni garanti ni configurable.

Si vous créez plusieurs hooks d'exécution du même type (par exemple, pre-snapshot), l'ordre d'exécution de ces hooks n'est pas garanti. Cependant, l'ordre d'exécution des hooks de types différents est garanti. Par exemple, voici l'ordre d'exécution d'une configuration qui comporte tous les différents types de hooks :

1. Hooks de pré-snapshot exécutés
2. Hooks post-snapshot exécutés
3. Hooks de pré-sauvegarde exécutés
4. Hooks post-sauvegarde exécutés

REMARQUE	L'exemple de commande précédent ne s'applique que lorsque vous exécutez une sauvegarde qui n'utilise pas d'instantané existant.
REMARQUE	Vous devez toujours tester vos scripts d'exécution hook avant de les activer dans un environnement de production. Vous pouvez utiliser la commande 'kubectl exec' pour tester facilement les scripts. Après avoir activé les hooks d'exécution dans un environnement de production, testez les snapshots et les sauvegardes résultants pour vous assurer qu'ils sont cohérents. Vous pouvez le faire en clonant l'application dans un espace de noms temporaire, en restaurant le snapshot ou la sauvegarde, puis en testant l'application.
REMARQUE	Si un hook d'exécution pré-snapshot ajoute, modifie ou supprime des ressources Kubernetes, ces modifications sont incluses dans le snapshot ou la sauvegarde et dans toute opération de restauration ultérieure.

Remarques importantes concernant les hooks d'exécution personnalisés

Tenez compte des éléments suivants lors de la planification des hooks d'exécution pour vos apps.

- Un point d'extension d'exécution doit utiliser un script pour effectuer des actions. De nombreux points d'extension d'exécution peuvent faire référence au même script.
- Trident Protect exige que les scripts que les hooks d'exécution utilisent soient écrits au format de scripts shell exécutables.
- La taille du script est limitée à 96KB.
- Trident Protect utilise les paramètres d'exécution de hook et tous les critères correspondants pour déterminer quels hooks sont applicables à une opération de snapshot, de sauvegarde ou de restauration.

REMARQUE	Parce que les hooks d'exécution réduisent souvent ou désactivent complètement la fonctionnalité de l'application sur laquelle ils s'exécutent, vous devez toujours essayer de minimiser le temps que vos hooks d'exécution personnalisés prennent pour s'exécuter. Si vous lancez une opération de sauvegarde ou de snapshot avec des hooks d'exécution associés, puis l'annulez, les hooks sont toujours autorisés à s'exécuter si l'opération de sauvegarde ou de snapshot a déjà commencé. Cela signifie que la logique utilisée dans un hook d'exécution post-sauvegarde ne peut pas supposer que la sauvegarde a été complétée.

Filtres de hooks d'exécution

Lorsque vous ajoutez ou modifiez un hook d'exécution pour une application, vous pouvez ajouter des filtres au hook d'exécution afin de gérer les conteneurs auxquels le hook correspond. Les filtres sont utiles pour les applications qui utilisent la même image de conteneur sur tous les conteneurs, mais peuvent utiliser chaque image à des fins différentes (comme Elasticsearch). Les filtres vous permettent de créer des scénarios où les hooks d'exécution s'exécutent sur certains, mais pas nécessairement tous, les conteneurs identiques. Si vous créez plusieurs filtres pour un seul hook d'exécution, ils sont combinés avec un opérateur logique AND. Vous

pouvez avoir jusqu'à 10 filtres actifs par hook d'exécution.

Chaque filtre que vous ajoutez à un hook d'exécution utilise une expression régulière pour faire correspondre les conteneurs dans 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 Regular Expression 2 (RE2), qui ne prend pas en charge la création d'un filtre excluant des conteneurs de la liste des correspondances. Pour obtenir des informations sur la syntaxe que Trident Protect prend en charge pour les expressions régulières dans les filtres de hooks d'exécution, consultez "Prise en charge de la syntaxe Regular Expression 2 (RE2)".

REMARQUE	Si vous ajoutez un filtre d'espace de noms à un hook d'exécution qui s'exécute après une opération de restauration ou de clonage et que la source et la destination de la restauration ou du clonage se trouvent dans des espaces de noms différents, le filtre d'espace de noms ne s'applique qu'à l'espace de noms de destination.

Exemples de hooks d'exécution

Visitez le "Projet GitHub NetApp Verda" pour télécharger des hooks d'exécution réels pour des applications populaires telles que Apache Cassandra et Elasticsearch. Vous pouvez également voir des exemples et obtenir des idées pour structurer vos propres hooks d'exécution personnalisés.

Créer un hook d'exécution

Vous pouvez créer un hook d'exécution personnalisé pour une application à l'aide de Trident Protect. Vous devez disposer des autorisations Owner, Admin ou Member pour créer des hooks d'exécution.

Utilisez un 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**: (*Obligatoire*) Le nom de cette ressource personnalisée; choisissez un nom unique et pertinent pour votre environnement.
   - **spec.applicationRef**: (*Obligatoire*) Le nom Kubernetes de l'application pour laquelle exécuter le hook d'exécution.
   - **spec.stage** : (*Obligatoire*) Chaîne de caractères indiquant à quelle étape de l'action le hook d'exécution doit s'exécuter. Valeurs possibles :
     - Pré
     - Publier
   - **spec.action** : (*Obligatoire*) Une chaîne indiquant l'action que le hook d'exécution effectuera, en supposant que les filtres de hook d'exécution spécifiés correspondent. Valeurs possibles :
     - Instantané
     - Sauvegarde
     - Restaurer
     - Basculement
   - **spec.enabled** : (*Optionnel*) Indique si ce hook d'exécution est activé ou désactivé. Si non spécifié, la valeur par défaut est true.
   - **spec.hookSource**: (*Obligatoire*) Une chaîne contenant le script de hook encodé en base64.
   - **spec.timeout** : (*Optionnel*) Un nombre définissant combien de temps, en minutes, le hook 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**: (*Optionnel*) Une liste YAML d'arguments que vous pouvez spécifier pour le hook d'exécution.
   - **spec.matchingCriteria** : (*Facultatif*) Liste facultative de paires clé-valeur de critères, chaque paire constituant un filtre d'exécution hook. Vous pouvez ajouter jusqu'à 10 filtres par exécution hook.
   - **spec.matchingCriteria.type** : (*Optionnel*) Une chaîne identifiant le type de filtre du hook d'exécution. Valeurs possibles :
     - ContainerImage
     - ContainerName
     - PodName
     - PodLabel
     - NamespaceName
   - **spec.matchingCriteria.value**: (*Optionnel*) Une chaîne de caractères ou une expression régulière identifiant la valeur du filtre du hook 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. Après avoir rempli le fichier CR avec les valeurs correctes, appliquez le CR :

```
kubectl apply -f trident-protect-hook.yaml
```

Utilisez la ligne de commandes (CLI)

Étapes

1. Créez le hook d'exécution en remplaçant les valeurs entre crochets 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>
```

Exécuter manuellement un hook d'exécution

Vous pouvez exécuter manuellement un script d'exécution à des fins de test ou si vous devez le relancer manuellement après un échec. Vous devez disposer des autorisations de propriétaire, d'administrateur ou de membre pour exécuter manuellement des scripts d'exécution.

L'exécution manuelle d'un hook consiste en deux étapes de base :

1. Créez une sauvegarde de ressources, qui collecte les ressources et crée une sauvegarde de celles-ci, déterminant où le hook s'exécutera
2. Exécutez le hook d'exécution sur la sauvegarde

Étape 1 : Créer une sauvegarde des ressources

Utilisez un 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**: (*Obligatoire*) Le nom de cette ressource personnalisée; choisissez un nom unique et pertinent pour votre environnement.
   - **spec.applicationRef**: (*Obligatoire*) Le nom Kubernetes de l'application pour laquelle créer la sauvegarde de ressources.
   - **spec.appVaultRef** : (*Obligatoire*) Le nom du AppVault où les contenus de sauvegarde sont stockés.
   - **spec.appArchivePath** : Le chemin à l'intérieur de AppVault où le contenu de la sauvegarde est stocké. 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. Après avoir rempli le fichier CR avec les valeurs correctes, appliquez le CR :

   ```
   kubectl apply -f trident-protect-resource-backup.yaml
   ```

Utilisez la ligne de commandes (CLI)

Étapes

1. Créez la sauvegarde en remplaçant les valeurs entre crochets 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. Consultez l'état de la sauvegarde. Vous pouvez utiliser cet exemple de commande à plusieurs reprises jusqu'à la fin de l'opération :

```
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 : Exécuter le hook d’exécution

Utilisez un 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**: (*Obligatoire*) Le nom de cette ressource personnalisée; choisissez un nom unique et pertinent pour votre environnement.
   - **spec.applicationRef**: (*Obligatoire*) Assurez-vous que cette valeur corresponde au nom de l'application de la ResourceBackup CR que vous avez créée à l'étape 1.
   - **spec.appVaultRef** : (*Obligatoire*) Assurez-vous que cette valeur corresponde à la appVaultRef du ResourceBackup CR que vous avez créée à l'étape 1.
   - **spec.appArchivePath** : Assurez-vous que cette valeur corresponde à la appArchivePath 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** : (*Obligatoire*) Une chaîne indiquant l'action que le hook d'exécution effectuera, en supposant que les filtres de hook d'exécution spécifiés correspondent. Valeurs possibles :
     - Instantané
     - Sauvegarde
     - Restaurer
     - Basculement
   - **spec.stage** : (*Obligatoire*) Chaîne de caractères indiquant à quelle étape de l'action le hook d'exécution doit s'exécuter. Ce hook run n'exécutera aucun hook à une autre étape. Valeurs possibles :
     - Pré
     - Publier

     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. Après avoir rempli le fichier CR avec les valeurs correctes, appliquez le CR :

```
kubectl apply -f trident-protect-hook-run.yaml
```

Utilisez la ligne de commandes (CLI)

Étapes

1. Créer la requête d'exécution manuelle du hook :

```
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 la fin de l'opération :

```
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>
```

Désinstaller Trident Protect

Vous devrez peut-être supprimer les composants Trident Protect si vous passez d'une version d'essai à une version complète du produit.

Pour supprimer Trident Protect, effectuez les étapes suivantes.

Étapes

1. Supprimez les fichiers CR de Trident Protect :

   REMARQUE Cette étape n'est pas requise pour la version 25.06 et 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 pouvez trouver d'excellents blogs sur NetApp Trident et Trident Protect ici :

Blogs de 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 for ONTAP avec l'add-on Amazon EKS"
- 15 avril 2025 : "NetApp Trident avec Google Cloud NetApp Volumes pour le protocole SMB"
- 14 avril 2025 : "Utilisation du protocole Fiber 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 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 SMB avec Google Cloud NetApp Volumes"
- 5 mars 2025 : "Débloquez l'intégration transparente du stockage iSCSI : Guide de FSxN sur les clusters ROSA 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 pour NetApp ONTAP avec Red Hat OpenShift Service sur AWS (ROSA) utilisant Trident"
- 29 octobre 2024 : "Migration à chaud de machines virtuelles avec OpenShift Virtualization sur ROSA et Amazon FSx pour 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 : "OpenShift Virtualization Reprise après sinistre avec NetApp Trident Protect"
- 13 mai 2025 : "Migration de classes de stockage avec Trident Protect sauvegarde et restauration"
- 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 en cas de panne pour les machines virtuelles OpenShift Virtualization"
- 11 mars 2025: "Extension des modèles GitOps à la protection des données applicatives avec NetApp Trident"
- 3 mars 2025: "Trident 25.02 : Amélioration de l'expérience Red Hat OpenShift avec de nouvelles fonctionnalités passionnantes"
- 15 janvier 2025 : "Présentation du contrôle d'accès basé sur les rôles de Trident Protect"
- 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 concernant l'installation, la configuration, la mise à niveau et le dépannage Trident.

Questions générales

À quelle fréquence Trident est-il publié ?

À compter de la version 24.02, Trident est publié tous les quatre mois : février, juin et octobre.

Trident prend-il en charge toutes les fonctionnalités publiées dans une version particulière de Kubernetes ?

Trident ne prend généralement pas en charge les fonctionnalités alpha de Kubernetes. Trident peut prendre en charge les fonctionnalités bêta dans les deux versions de Trident qui suivent la version bêta de Kubernetes.

Trident a-t-il des dépendances à d'autres produits NetApp pour son fonctionnement ?

Trident ne dépend d'aucun autre produit logiciel NetApp et fonctionne comme une application autonome. Cependant, vous devez disposer d'un périphérique de stockage NetApp.

Comment puis-je obtenir tous les détails de configuration de Trident ?

Utilisez la commande `tridentctl get` pour obtenir plus d'informations sur votre configuration Trident.

Puis-je obtenir des métriques sur la manière dont le stockage est provisionné par Trident ?

Oui. Les points de terminaison Prometheus peuvent être utilisés pour recueillir des informations sur le fonctionnement de Trident, telles que le nombre de backends gérés, le nombre de volumes provisionnés, les octets consommés, etc. Vous pouvez également utiliser "Cloud Insights" pour la surveillance et l'analyse.

L'expérience utilisateur change-t-elle lors de l'utilisation de Trident en tant que CSI Provisioner ?

Non. L'expérience utilisateur et les fonctionnalités restent inchangées. Le nom du provisionneur utilisé est `csi.trident.netapp.io`. Cette méthode d'installation de Trident est recommandée si vous souhaitez utiliser toutes les nouvelles fonctionnalités des versions actuelles et futures.

Installer et utiliser 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 "Découvrez l'installation de Trident".

Puis-je installer Trident à distance ?

Oui. Trident 18.10 et versions ultérieures prennent en charge la capacité d'installation à distance depuis n'importe quelle machine qui a `kubectl` accès au cluster. Après que `kubectl` l'accès a été vérifié (par exemple, initiez une `kubectl get nodes` commande depuis la machine distante 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 seule instance, ce qui lui confère une haute disponibilité intégrée. Vous ne devez pas augmenter le nombre de réplicas dans le déploiement. Si le nœud où Trident est installé est perdu ou si le pod est autrement inaccessible, Kubernetes redéploie automatiquement le pod sur un nœud sain de votre cluster. Trident est uniquement un composant du plan de contrôle, donc les pods actuellement montés ne sont pas affectés si Trident est redéployé.

Trident a-t-il besoin d'accéder à l'espace de noms kube-system ?

Trident lit à partir du serveur d'API Kubernetes pour déterminer quand les applications demandent de nouveaux PVC, il a donc besoin d'accéder à kube-system.

Quels sont les rôles et privilèges 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. Consultez "Personnaliser l'installation de tridentctl".

Puis-je générer localement les fichiers manifestes 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. Consultez "Personnaliser l'installation de tridentctl".

Puis-je partager le même backend SVM ONTAP pour deux instances Trident distinctes pour deux clusters Kubernetes distincts ?

Bien que cela soit déconseillé, vous pouvez utiliser le même SVM backend pour deux instances Trident. Spécifiez un nom de volume unique pour chaque instance lors de l'installation et/ou indiquez un paramètre unique `StoragePrefix` dans le fichier `setup/backend.json`. Cela permet de garantir que le même volume FlexVol n'est pas utilisé pour les deux instances.

Est-il possible d'installer Trident sous ContainerLinux (anciennement CoreOS) ?

Trident est simplement un pod Kubernetes et peut être installé partout où Kubernetes est en cours d'exécution.

Puis-je utiliser Trident avec NetApp Cloud Volumes ONTAP ?

Oui, Trident est pris en charge sur AWS, Google Cloud et Azure.

Dépannage et support

Est-ce que NetApp prend en charge Trident ?

Bien que Trident soit open source et fourni gratuitement, NetApp le prend entièrement en charge à condition que votre backend NetApp soit pris en charge.

Comment puis-je ouvrir un dossier d'assistance

Pour ouvrir un dossier de demande d'assistance, effectuez l'une des opérations suivantes :

1. Contactez votre Support Account Manager et obtenez de l'aide pour créer un ticket.
2. Ouvrez un ticket d'assistance en contactant "Support NetApp".

Comment générer un bundle de journal de support ?

Vous pouvez créer un bundle de support en exécutant `tridentctl logs -a`. En plus des journaux inclus dans le bundle, capturez le journal kubelet pour diagnostiquer les problèmes de montage côté Kubernetes. Les instructions pour obtenir le journal kubelet varient selon la façon dont Kubernetes est installé.

Que dois-je faire si je dois soumettre une demande pour une nouvelle fonctionnalité ?

Créez un ticket sur "Trident Github" et mentionnez **RFE** dans l'objet et la description du problème.

Où dois-je signaler un défaut ?

Créez un ticket sur "Trident Github". Veillez à inclure toutes les informations et les journaux nécessaires relatifs à ce problème.

Que se passe-t-il si j'ai une question rapide sur Trident et que j'ai besoin d'éclaircissements ? Existe-t-il une communauté ou un forum ?

Si vous avez des questions, des problèmes ou des demandes, contactez-nous via notre Trident "Chaîne Discord" 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 backend avec `tridentctl update backend myBackend -f </path/to_new_backend.json> -n trident`. Remplacez `myBackend` dans l'exemple par le nom de votre backend, et `/path/to_new_backend.json` par le chemin d'accès au fichier `backend.json` correct.

Trident ne parvient pas à trouver mon nœud Kubernetes. Comment puis-je résoudre ce problème ?

Il existe deux scénarios probables pour lesquels Trident ne peut pas trouver un nœud Kubernetes. Cela peut être dû à un problème de réseau au sein de Kubernetes ou à un problème de DNS. Le daemonset de nœud Trident qui s'exécute sur chaque nœud Kubernetes doit pouvoir communiquer avec le contrôleur Trident pour enregistrer le nœud auprès de Trident. Si des modifications réseau sont survenues après l'installation de Trident, vous ne rencontrez ce problème qu'avec les nouveaux nœuds Kubernetes ajoutés au cluster.

Si le pod Trident est détruit, vais-je perdre les données ?

Les données ne seront pas perdues si le pod Trident est détruit. Les métadonnées Trident sont stockées dans des objets CRD. Tous les PV qui ont été provisionnés par Trident fonctionneront normalement.

Mettre à niveau Trident

Puis-je passer directement d'une version antérieure à une version plus récente (en sautant quelques versions) ?

NetApp prend en charge la mise à niveau de Trident d'une version majeure à la version majeure suivante. Vous pouvez effectuer une mise à niveau de la version 18.xx à 19.xx, de 19.xx à 20.xx, et ainsi de suite. Vous devez tester la mise à niveau dans un environnement de test avant le déploiement en production.

Est-il possible de rétrograder Trident vers une version précédente ?

Si vous avez besoin d'un correctif pour des bugs observés après une mise à niveau, des problèmes de dépendances ou une mise à niveau infructueuse ou incomplète, vous devez "désinstaller Trident" et réinstaller

la version précédente en utilisant les instructions spécifiques à cette version. C'est la seule méthode recommandée pour revenir à une version antérieure.

Gérer les backends et les volumes

Dois-je définir à la fois les LIF de gestion et de données dans un fichier de définition backend ONTAP ?

Le LIF de gestion est obligatoire. Le DataLIF varie :

- ONTAP SAN : Ne spécifiez pas pour iSCSI. Trident utilise "Mappage LUN sélectif ONTAP" pour découvrir les LIF iSCSI nécessaires à l'établissement d'une session multi-chemin. Un avertissement est généré si `dataLIF` est explicitement défini. Consultez "Options et exemples de configuration SAN ONTAP" pour plus de détails.
- ONTAP NAS : 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 complet (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. Consultez "Options et exemples de configuration NAS ONTAP" pour plus de détails

Trident peut-il configurer CHAP pour les backends ONTAP ?

Oui. Trident prend en charge le protocole CHAP bidirectionnel pour les backends ONTAP. Cela nécessite de définir `useCHAP=true` dans la configuration de votre backend.

Comment gérer les règles d'export avec Trident ?

Trident peut créer et gérer dynamiquement des règles d'export à partir de la version 20.04. Cela permet à l'administrateur de stockage de fournir un ou plusieurs blocs CIDR dans sa configuration backend et à Trident d'ajouter les adresses IP des nœuds qui se trouvent dans ces plages à une règle d'export qu'il crée. De cette manière, Trident gère automatiquement l'ajout et la suppression des règles pour les nœuds dont les adresses IP se trouvent dans les CIDR donnés.

Les adresses IPv6 peuvent-elles être utilisées pour les Management et DataLIFs ?

Trident prend en charge la définition d'adresses IPv6 pour :

- `managementLIF` et `dataLIF` pour les backends NAS ONTAP.
- `managementLIF` pour les backends SAN ONTAP. Vous ne pouvez pas spécifier `dataLIF` sur un backend SAN ONTAP.

Trident doit être installé en utilisant le flag `--use-ipv6` (pour `tridentctl` l'installation), `IPv6` (pour Trident operator), ou `tridentTPv6` (pour l'installation Helm) pour qu'il fonctionne sur IPv6.

Est-il possible de mettre à jour le LIF de gestion sur le backend ?

Oui, il est possible de mettre à jour le LIF de gestion backend à l'aide de la commande `tridentctl update backend`.

Est-il possible de mettre à jour le DataLIF sur le backend ?

Vous pouvez mettre à jour le DataLIF sur `ontap-nas` et `ontap-nas-economy` uniquement.

Puis-je créer plusieurs backends dans Trident pour Kubernetes ?

Trident peut prendre en charge simultanément de nombreux backends, soit avec le même pilote, soit avec des pilotes différents.

Comment Trident stocke-t-il les identifiants du backend ?

Trident stocke les identifiants du backend sous forme de Secrets Kubernetes.

Comment Trident sélectionne-t-il un backend spécifique ?

Si les attributs du backend ne peuvent pas être utilisés pour sélectionner automatiquement les pools appropriés pour une classe, les `storagePools` et `additionalStoragePools` paramètres sont utilisés pour sélectionner un ensemble spécifique de pools.

Comment puis-je m'assurer que Trident ne sera pas provisionné à partir d'un backend spécifique ?

Le `excludeStoragePools` paramètre est utilisé pour filtrer l'ensemble des pools que Trident utilise pour le provisionnement et supprimera tous les pools correspondants.

S'il existe plusieurs backends du même type, comment Trident sélectionne-t-il celui à utiliser ?

S'il existe plusieurs backends configurés du même type, Trident sélectionne le backend approprié en fonction des paramètres présents dans `StorageClass` et `PersistentVolumeClaim`. Par exemple, s'il existe plusieurs backends de pilote ontap-nas, Trident tente de faire correspondre les paramètres dans le `StorageClass` et `PersistentVolumeClaim` combinés et de trouver un backend capable de répondre aux exigences listées dans `StorageClass` et `PersistentVolumeClaim`. Si plusieurs backends correspondent à la requête, Trident en sélectionne un au hasard.

Trident prend-il en charge le protocole CHAP bidirectionnel avec Element/SolidFire ?

Oui.

Comment Trident déploie-t-il des Qtrees sur un volume ONTAP ? Combien de Qtrees peuvent être déployés sur un seul volume ?

Le `ontap-nas-economy` pilote crée jusqu'à 200 Qtrees dans le même volume FlexVol (configurable entre 50 et 300), 100 000 Qtrees par nœud de cluster et 2,4 M par cluster. Lorsque vous saisissez un nouveau `PersistentVolumeClaim` qui est pris en charge par le pilote économique, le pilote vérifie si un volume FlexVol existe déjà et peut prendre en charge le nouveau Qtree. Si le volume FlexVol n'existe pas pour prendre en charge le Qtree, un nouveau volume FlexVol est créé.

Comment puis-je configurer les permissions Unix pour les volumes provisionnés sur ONTAP NAS ?

Vous pouvez définir les permissions Unix sur le volume provisionné par Trident en définissant un paramètre dans le fichier de définition du backend.

Comment puis-je configurer un ensemble explicite d'options de montage NFS ONTAP lors du provisionnement d'un volume ?

Par défaut, Trident ne configure aucune option de montage avec Kubernetes. Pour spécifier les options de montage dans la classe de stockage Kubernetes, suivez l'exemple fourni "ici".

Comment puis-je configurer les volumes provisionnés selon une règle d'export ?

Pour autoriser les hôtes concernés à accéder à un volume, utilisez le `exportPolicy` paramètre configuré dans le fichier de définition du backend.

Comment configurer le chiffrement du volume via Trident avec ONTAP ?

Vous pouvez configurer le chiffrement sur le volume provisionné par Trident en utilisant le paramètre encryption dans le fichier de définition du backend. Pour plus d'informations, consultez : "Comment Trident fonctionne avec NVE et NAE"

Quelle est la meilleure façon de mettre en œuvre la QoS pour ONTAP via Trident ?

Utilisez `StorageClasses` pour implémenter la QoS pour ONTAP.

Comment spécifier un provisionnement fin ou épais via Trident ?

Les pilotes ONTAP prennent en charge le provisionnement fin ou épais. Les pilotes ONTAP utilisent par défaut le provisionnement fin. Si un provisionnement épais est souhaité, vous devez configurer soit le fichier de définition du backend, soit le `StorageClass`. Si les deux sont configurés, `StorageClass` prend le dessus. Configurez ce qui suit pour ONTAP :

1. Sur `StorageClass`, définissez l'attribut `provisioningType` sur épais.
2. Dans le fichier de définition du backend, activez les volumes épais en définissant `backend spaceReserve parameter` comme volume.

Comment puis-je m'assurer que les volumes utilisés ne sont pas supprimés même si je supprime accidentellement le PVC ?

La protection PVC est automatiquement activée sur Kubernetes à partir de la version 1.10.

Puis-je augmenter la taille des PVC NFS créés par Trident ?

Oui. Vous pouvez étendre un PVC créé par Trident. Notez que l'autogrow de volume est une fonctionnalité ONTAP qui ne s'applique pas à Trident.

Puis-je importer un volume alors qu'il est 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 le quota de ressources est-il traduit dans un cluster NetApp ?

Le quota de ressources de stockage Kubernetes doit fonctionner tant que le stockage NetApp dispose de capacité. Lorsque le stockage NetApp ne peut pas respecter les paramètres de quota Kubernetes en raison

d'un manque de capacité, Trident essaie de provisionner mais échoue.

Puis-je créer des instantanés de volume avec Trident ?

Oui. La création d'instantanés de volumes à la demande et de volumes persistants à partir d'instantanés est prise en charge par Trident. Pour créer des volumes persistants (PVs) à partir d'instantanés, assurez-vous que la `VolumeSnapshotDataSource` feature gate a été activée.

Quels sont les pilotes qui prennent en charge les instantanés de volume Trident ?

À ce jour, la prise en charge des instantanés à la demande est disponible pour nos `ontap-nas`, `ontap-nas-flexgroup`, `ontap-san`, `ontap-san-economy`, `solidfire-san` et `azure-netapp-files` pilotes backend.

Comment puis-je effectuer une sauvegarde instantanée d'un volume provisionné par Trident avec ONTAP ?

Cette fonctionnalité est disponible sur `ontap-nas`, `ontap-san`, et `ontap-nas-flexgroup` pilotes. Vous pouvez également spécifier un `snapshotPolicy` pour le `ontap-san-economy` pilote au niveau FlexVol.

Cette fonctionnalité est également disponible sur les `ontap-nas-economy` pilotes, mais avec une granularité au niveau du volume FlexVol et non au niveau de la granularité du qtree. Pour activer la possibilité de créer des instantanés des volumes provisionnés par Trident, définissez l'option du paramètre backend `snapshotPolicy` sur la stratégie d'instantané souhaitée telle que définie sur le backend ONTAP. Les instantanés créés par le contrôleur de stockage ne sont pas connus de Trident.

Puis-je définir un pourcentage de réserve de snapshot pour un volume provisionné via Trident ?

Oui, vous pouvez réserver un pourcentage spécifique d'espace disque pour le stockage des copies de snapshots via Trident en définissant l'attribut `snapshotReserve` dans le fichier de définition du backend. Si vous avez configuré `snapshotPolicy` et `snapshotReserve` dans le fichier de définition du backend, le pourcentage de réservation des snapshots est défini selon le pourcentage `snapshotReserve` mentionné dans le fichier backend. Si le nombre de pourcentage `snapshotReserve` n'est pas mentionné, ONTAP prend par défaut le pourcentage de réservation des snapshots comme 5. Si l'option `snapshotPolicy` est définie sur none, le pourcentage de réservation des snapshots est fixé à 0.

Puis-je accéder directement au répertoire des snapshots de volume et copier des fichiers ?

Oui, vous pouvez accéder au répertoire des instantanés sur le volume provisionné par Trident en configurant le `snapshotDir` paramètre dans le fichier de définition du backend.

Puis-je configurer SnapMirror pour les volumes via Trident ?

Actuellement, SnapMirror doit être configuré en externe en utilisant l'interface de ligne de commande ONTAP ou OnCommand System Manager.

Comment restaurer des volumes persistants à un instantané ONTAP spécifique ?

Pour restaurer un volume à un snapshot ONTAP, procédez comme suit :

1. Mettez en veille le pod d'application qui utilise le volume Persistent.
2. Restaurez l'instantané requis via l'interface de ligne de commande ONTAP ou OnCommand System Manager.

3. Redémarrez le pod d'application.

Trident peut-il provisionner des volumes sur des SVM ayant un miroir de partage de charge configuré ?

Des miroirs de répartition de charge peuvent être créés pour les volumes racine des SVM qui diffusent des données via NFS. ONTAP met automatiquement à jour les miroirs de répartition de charge pour les volumes créés par Trident. Cela peut entraîner des délais lors du montage des volumes. Lorsque plusieurs volumes sont créés avec Trident, le provisionnement d'un volume dépend de la mise à jour du miroir de répartition de charge par ONTAP.

Comment puis-je séparer l'utilisation des classes de stockage pour chaque client/locataire ?

Kubernetes n'autorise pas les classes de stockage dans les espaces de noms. Cependant, vous pouvez utiliser Kubernetes pour limiter l'utilisation d'une classe de stockage spécifique par espace de noms en utilisant les quotas de ressources de stockage, qui sont définis par espace de noms. Pour refuser à un espace de noms spécifique l'accès à un stockage spécifique, définissez le quota de ressources de cette classe de stockage à 0.

Dépannage

Utilisez les indications fournies ici pour résoudre les problèmes que vous pourriez rencontrer lors de l'installation et de l'utilisation de Trident.

REMARQUE	Pour obtenir de l'aide concernant Trident, créez un dossier de support à l'aide de `tridentctl logs -a -n trident` et envoyez-le à NetApp Support.

Dépannage général

- Si le pod Trident ne démarre pas correctement (par exemple, lorsque le pod Trident est bloqué dans la `ContainerCreating` phase avec moins de deux conteneurs prêts), exécuter `kubectl -n trident describe deployment trident` et `kubectl -n trident describe pod trident-**` peut fournir des informations supplémentaires. L'obtention des journaux kubelet (par exemple, via `journalctl -xeu kubelet`) peut également être utile.
- S'il n'y a pas assez d'informations dans les journaux Trident, vous pouvez essayer d'activer le mode débogage pour Trident en passant le `-d` indicateur au paramètre d'installation selon votre option d'installation.

  Ensuite, confirmez que le mode débogage est activé en utilisant `./tridentctl logs -n trident` et en recherchant `level=debug msg` dans le journal.

  Installé avec Operator

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

  Cela redémarrera tous les pods Trident, ce qui peut prendre plusieurs secondes. Vous pouvez le vérifier en observant la colonne « AGE » dans la sortie de `kubectl get pod -n trident`.

  Pour Trident 20.07 et 20.10, utilisez `tprov` à la place de `torc`.

Installé avec Helm

```
helm upgrade <name> trident-operator-21.07.1-custom.tgz --set
tridentDebug=true`
```

Installé avec tridentctl

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

- Vous pouvez également obtenir les journaux de débogage pour chaque backend en incluant `debugTraceFlags` dans la définition de votre backend. Par exemple, incluez `debugTraceFlags: {"api":true, "method":true,}` pour obtenir les appels d'API et les parcours de méthodes dans les journaux Trident. Les backends existants peuvent avoir `debugTraceFlags` configuré avec un `tridentctl backend update`.
- Lors de l'utilisation de Red Hat Enterprise Linux CoreOS (RHCOS), assurez-vous que `iscsid` est activé sur les nœuds de travail et démarré par défaut. Cela peut être fait en utilisant OpenShift MachineConfigs ou en modifiant les modèles d'ignition.
- 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 lors du montage d'un PV dans un conteneur, assurez-vous que `rpcbind`est installé et en cours d'exécution. Utilisez le gestionnaire de paquets requis pour le système d'exploitation hôte et vérifiez si `rpcbind`est en cours d'exécution. Vous pouvez vérifier l'état du `rpcbind`service en exécutant un `systemctl status rpcbind`ou une commande équivalente.
- Si un backend Trident signale qu'il est dans l'état `failed` alors qu'il fonctionnait auparavant, cela est probablement dû à une modification des identifiants SVM/admin associés au backend. La mise à jour des informations du backend à l'aide de `tridentctl update backend` ou le redémarrage du pod Trident résoudra ce problème.
- Si vous rencontrez des problèmes d'autorisation lors de l'installation de Trident avec Docker comme environnement d'exécution de conteneurs, essayez d'installer Trident avec le `--in cluster=false` flag. Cela n'utilisera pas de pod d'installation et évitera les problèmes d'autorisation rencontrés à cause de l'utilisateur `trident-installer`.
- Utilisez la `uninstall parameter <Uninstalling Trident>` pour nettoyer après une exécution ayant échoué. Par défaut, le script ne supprime pas les CRDs créées par Trident, ce qui permet de désinstaller et de réinstaller en toute sécurité, même dans un déploiement en cours d'exécution.
- Si vous souhaitez revenir à une version antérieure de Trident, commencez par exécuter la commande `tridentctl uninstall` pour supprimer Trident. Téléchargez la version souhaitée "Version Trident" et installez-la à l'aide de la commande `tridentctl install`.
- Après une installation réussie, si un PVC est bloqué dans la `Pending` phase, exécuter `kubectl describe pvc` peut fournir des informations supplémentaires sur la raison pour laquelle Trident n'a pas pu provisionner un PV pour ce PVC.

Déploiement Trident infructueux avec l'opérateur

Si vous déployez Trident à l'aide de l'opérateur, l'état `TridentOrchestrator` change de `Installing` à

`Installed`. Si vous observez le statut `Failed` et que l'opérateur ne parvient pas à se rétablir automatiquement, vous devez vérifier les journaux de l'opérateur en exécutant la commande suivante :

```
tridentctl logs -l trident-operator
```

L'analyse des journaux du conteneur trident-operator peut permettre d'identifier l'origine du problème. Par exemple, l'une de ces difficultés pourrait être l'impossibilité de récupérer les images de conteneur requises auprès des registres en amont dans un environnement airgapped.

Pour comprendre pourquoi l'installation de Trident a échoué, vous devriez consulter l' `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 une `TridentOrchestrator` qui a été utilisée pour installer Trident. Étant donné que chaque cluster Kubernetes ne peut avoir qu'une seule instance de Trident, l'opérateur s'assure qu'à tout moment, il n'existe qu'une seule `TridentOrchestrator` active qu'il peut créer.

De plus, l'observation de l'état des pods Trident peut souvent indiquer si quelque chose ne va pas.

```
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
```

On constate clairement que les pods ne peuvent pas s'initialiser complètement car une ou plusieurs images de conteneur n'ont pas été récupérées.

Pour résoudre le problème, vous devez modifier la `TridentOrchestrator` CR. Vous pouvez également supprimer `TridentOrchestrator`, et en créer une nouvelle avec la définition modifiée et correcte.

Déploiement Trident infructueux utilisant `tridentctl`

Pour tenter de comprendre ce qui s'est mal passé, vous pouvez relancer le programme d'installation en utilisant l'argument `-d`, ce qui activera le mode débogage et vous aidera à comprendre quel est le problème :

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

Après avoir résolu le problème, vous pouvez nettoyer l'installation comme suit, puis exécuter la commande `tridentctl install` 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.
```

Supprimez complètement Trident et les CRD

Vous pouvez supprimer complètement Trident ainsi que toutes les CRDs créées et les ressources personnalisées associées.

AVERTISSEMENT	Cette opération est irréversible. Ne la réalisez pas à moins de souhaiter une réinstallation complète de Trident. Pour désinstaller Trident sans supprimer les CRD, consultez "Désinstaller Trident".

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}}'
```

Helm

Pour désinstaller Trident et supprimer complètement les CRD à l'aide de Helm :

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

<code>tridentctl</code>

Pour supprimer complètement les CRD après la désinstallation de Trident en utilisant `tridentctl`

```
tridentctl obliviate crd
```

Échec du déstockage du nœud NVMe avec les espaces de noms de blocs bruts RWX sous Kubernetes 1.26

Si vous utilisez Kubernetes 1.26, la suppression des nœuds peut échouer lors de l'utilisation de NVMe/TCP avec des espaces de noms de blocs bruts RWX. Les scénarios suivants proposent des solutions de contournement à cette défaillance. Vous pouvez également mettre à niveau Kubernetes vers 1.27.

Suppression de l'espace de noms et du pod

Considérez un scénario où vous avez un espace de noms géré par Trident (volume persistant NVMe) attaché à un pod. Si vous supprimez l'espace de noms directement depuis le backend ONTAP, le processus de désactivation se bloque après que vous ayez tenté de supprimer le pod. Ce scénario n'affecte pas le cluster Kubernetes ni les autres fonctionnalités.

Solution de contournement

Démontez le volume persistant (correspondant à cet espace de noms) du nœud respectif et supprimez-le.

LIFs de données 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
Activez les dataLIFS pour rétablir la pleine fonctionnalité.
```

Suppression du mappage d'espace de noms

```
 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` au sous-système.
```

Les clients NFSv4.2 signalent « invalid argument » après la mise à niveau ONTAP alors qu'ils s'attendent à ce que « v4.2-xattrs » soit activé

Après la mise à niveau de ONTAP, les clients NFSv4.2 peuvent signaler des erreurs « argument invalide » lors de la tentative de montage des exports NFSv4.2. Ce problème survient lorsque l' `v4.2-xattrs`option n'est pas activée sur le SVM. Solution de contournement : activez l' `v4.2-xattrs`option sur le SVM ou mettez à niveau vers ONTAP 9.12.1 ou une version ultérieure, où cette option est activée par défaut.

Support

NetApp propose une assistance pour Trident de diverses manières. De nombreuses options d'auto-assistance gratuites sont disponibles 24h/24 et 7j/7, telles que des articles de la base de connaissances (KB) et un canal Discord.

Cycle de vie du support Trident

Trident propose trois niveaux d'assistance selon votre version. Consultez "NetApp Prise en charge des versions logicielles pour les définitions".

Soutien complet
: Trident offre un support complet pendant douze mois à compter de la date de sortie.

Support limité
: Trident offre un support limité pour les mois 13 à 24 à partir de la date de sortie.

Autonomie
: La documentation Trident est disponible pour les mois 25 à 36 à compter de la date de sortie.

Version	Soutien complet	Support limité	Autonomie
"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

Autonomie

Pour obtenir une liste complète des articles de dépannage, consultez "NetApp Knowledgebase (connexion requise)".

soutien communautaire

Il existe une communauté publique dynamique d'utilisateurs de conteneurs (y compris les développeurs de Trident) sur notre "Chaîne Discord". C'est un excellent endroit pour poser des questions générales sur le projet et discuter de sujets connexes avec des pairs partageant les mêmes idées.

Support technique NetApp

Pour obtenir de l'aide concernant Trident, créez un dossier de support à l'aide de `tridentctl logs -a -n trident` et envoyez-le à `NetApp Support <Getting Help>`.

Pour plus d'informations

- "Ressources Trident"
- "Hub Kubernetes"

Référence

Ports Trident

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

Aperçu

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

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

Ports pour les nœuds de contrôleur

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

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

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

Ports pour les nœuds de travail

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

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

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

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

API REST Trident

Bien que "commandes et options tridentctl" soit le moyen le plus simple d'interagir avec l'API REST de Trident, vous pouvez utiliser directement le point de terminaison REST si vous préférez.

Quand utiliser l'API REST

L'API REST est utile pour les installations avancées qui utilisent Trident comme binaire autonome dans des déploiements non-Kubernetes.

Pour une meilleure sécurité, le Trident `REST API` est limité par défaut à localhost lorsqu'il s'exécute dans un pod. Pour modifier ce comportement, vous devez définir l'argument de Trident `-address` dans la configuration de son pod.

Utilisation de l'API REST

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

L'API fonctionne comme suit :

GET

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

Liste tous les objets de ce type.

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

Obtient les détails de l'objet nommé.

POST

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

Crée un objet du type spécifié.

- Nécessite une configuration JSON pour que l'objet soit créé. Pour la spécification de chaque type d'objet, reportez-vous à "Gérez Trident à l'aide de tridentctl".
- Si l'objet existe déjà, le comportement varie : les backends mettent à jour l'objet existant, tandis que tous les autres types d'objet échoueront lors de l'opération.

SUPPRIMER

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

Supprime la ressource nommée.

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

Options de ligne de commande

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

Journalisation

`-debug`

Active la sortie de débogage.

-loglevel <level>

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

Kubernetes

-k8s_pod

Utilisez cette option ou `-k8s_api_server` pour activer la prise en charge de Kubernetes. En la configurant, Trident utilise les identifiants du compte de service Kubernetes du pod qui le contient pour contacter le serveur d'API. Cela ne fonctionne que lorsque Trident s'exécute en tant que pod dans un cluster Kubernetes avec les comptes de service activés.

-k8s_api_server <insecure-address:insecure-port>

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

Docker

-volume_driver <name>

Nom du pilote utilisé lors de l'enregistrement du plugin Docker. La valeur par défaut est `netapp`.

-driver_port <port-number>

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

-config <file>

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

REST

-address <ip-or-host>

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

AVERTISSEMENT	L'interface REST de Trident peut être configurée pour écouter et servir uniquement à 127.0.0.1 (pour IPv4) ou [::1] (pour IPv6).

-port <port-number>

Spécifie le port sur lequel le serveur REST de Trident doit écouter. Par défaut 8000.

-rest

Active l'interface REST. La valeur par défaut est true.

Objets Kubernetes et Trident

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

Comment les objets interagissent-ils entre eux ?

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

1. Un utilisateur crée un `PersistentVolumeClaim` en demandant un nouveau `PersistentVolume` d'une taille particulière à partir d'un Kubernetes `StorageClass` qui a été préalablement configuré par l'administrateur.
2. Le Kubernetes `StorageClass` identifie Trident comme son provisionneur et inclut des paramètres qui indiquent à Trident comment provisionner un volume pour la classe demandée.
3. Trident examine son propre `StorageClass` portant le même nom qui identifie la correspondance `Backends` et `StoragePools` qu'il peut utiliser pour provisionner des volumes pour la classe.
4. Trident provisionne le stockage sur un backend correspondant et crée deux objets : un `PersistentVolume` dans Kubernetes qui indique à Kubernetes comment trouver, monter et traiter le volume, et un volume dans Trident qui conserve la relation entre le `PersistentVolume` et le stockage réel.
5. Kubernetes associe le `PersistentVolumeClaim` au nouveau `PersistentVolume`. Les Pods qui incluent le `PersistentVolumeClaim` montent ce PersistentVolume sur n'importe quel hôte sur lequel ils s'exécutent.
6. Un utilisateur crée un `VolumeSnapshot` d'un PVC existant, en utilisant un `VolumeSnapshotClass` qui pointe vers Trident.
7. Trident identifie le volume associé au PVC et crée un instantané du volume sur son backend. Il crée également un `VolumeSnapshotContent` qui indique à Kubernetes comment identifier l'instantané.
8. Un utilisateur peut créer un `PersistentVolumeClaim` en utilisant `VolumeSnapshot` comme source.
9. Trident identifie l'instantané requis et effectue la même série d'étapes impliquées dans la création d'un `PersistentVolume` et d'un `Volume`.

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

Objets `PersistentVolumeClaim` Kubernetes

Un objet `PersistentVolumeClaim`Kubernetes est une demande de stockage effectuée par un utilisateur d'un cluster Kubernetes.

En plus de la spécification standard, Trident permet aux utilisateurs de spécifier les annotations spécifiques au volume suivantes s'ils souhaitent remplacer les valeurs par défaut que vous avez définies dans la configuration du backend :

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

Si le PV créé a la `Delete` stratégie de récupération, Trident supprime à la fois le PV et le volume sous-jacent lorsque le PV est libéré (c'est-à-dire lorsque l'utilisateur supprime le PVC). Si l'action de suppression échoue, Trident marque le PV comme tel et réessaie périodiquement l'opération jusqu'à ce qu'elle réussisse ou que le PV soit supprimé manuellement. Si le PV utilise la `Retain` stratégie, Trident l'ignore et suppose que l'administrateur le supprimera de Kubernetes et du backend, permettant ainsi de sauvegarder ou d'inspecter le volume avant sa suppression. Notez que la suppression du PV n'entraîne pas la suppression du volume sous-jacent par Trident. Vous devez le supprimer à l'aide de l'API REST (`tridentctl`.

Trident prend en charge la création d'instantanés de volumes à l'aide de la spécification CSI : vous pouvez créer un instantané de volume et l'utiliser comme source de données pour cloner des PVC existants. Ainsi, des copies à un instant donné des PV peuvent être exposées à Kubernetes sous forme d'instantanés. Les instantanés peuvent ensuite être utilisés pour créer de nouveaux PV. Consultez `On-Demand Volume Snapshots` pour voir comment cela fonctionne.

Trident fournit également les `cloneFromPVC` et `splitOnClone` annotations pour la création de clones. Vous pouvez utiliser ces annotations pour cloner un PVC sans avoir recours à l'implémentation CSI.

Voici un exemple : si un utilisateur possède déjà un PVC appelé `mysql`, l'utilisateur peut créer un nouveau PVC appelé `mysqlclone` en utilisant l'annotation, telle que `trident.netapp.io/cloneFromPVC: mysql`. Avec cette annotation définie, Trident clone le volume correspondant au PVC mysql, au lieu de provisionner un volume de zéro.

Considérez les points suivants :

- NetApp recommande de cloner un volume inactif.
- Un PVC et son clone doivent se trouver dans le même espace de noms Kubernetes et avoir la même classe de stockage.
- Avec les `ontap-nas` et `ontap-san` pilotes, il peut être souhaitable de définir l'annotation PVC `trident.netapp.io/splitOnClone` en conjonction avec `trident.netapp.io/cloneFromPVC`. Avec `trident.netapp.io/splitOnClone` défini sur `true`, Trident sépare le volume cloné du volume parent, découplant ainsi complètement le cycle de vie du volume cloné de celui de son parent, au prix d'une perte d'efficacité de stockage. Ne pas définir `trident.netapp.io/splitOnClone` ou le définir sur `false` entraîne une réduction de la consommation d'espace sur le backend, mais crée des dépendances entre les volumes parent et clone, de sorte que le volume parent ne peut pas être supprimé à moins que le clone ne le soit d'abord. Un scénario où la séparation du clone a du sens est le clonage d'un volume de base de données vide, lorsqu'on s'attend à ce que le volume et son clone divergent fortement et ne bénéficient pas des gains d'efficacité de stockage offerts par ONTAP.

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

Objets `PersistentVolume` Kubernetes

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

REMARQUE	Trident crée `PersistentVolume` des objets et les enregistre automatiquement auprès du cluster Kubernetes en fonction des volumes qu'il provisionne. Vous n'êtes pas censé les gérer vous-même.

Lorsque vous créez un PVC faisant référence à un volume basé sur Trident `StorageClass`, Trident provisionne un nouveau volume en utilisant la classe de stockage correspondante et enregistre un nouveau PV pour ce volume. Lors de la configuration du volume provisionné et du PV correspondant, Trident suit les règles suivantes :

- Trident génère un nom de PV pour Kubernetes et un nom interne qu'il utilise pour provisionner le stockage. Dans les deux cas, il s'assure que les noms sont uniques dans leur portée.
- La taille du volume correspond au mieux à la taille demandée dans le PVC, bien qu'elle puisse être arrondie à la quantité allouable la plus proche, selon la plateforme.

Objets `StorageClass` Kubernetes

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

Il s'agit de l'un des deux objets de base que l'administrateur doit créer et gérer. L'autre est l'objet backend Trident.

Un objet Kubernetes `StorageClass` utilisant Trident ressemble à ceci :

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: <Name>
provisioner: csi.trident.netapp.io
mountOptions: <Mount Options>
parameters: <Trident Parameters>
allowVolumeExpansion: true
volumeBindingMode: Immediate
```

Ces paramètres sont spécifiques à Trident et indiquent à Trident comment provisionner les volumes pour la classe.

Les paramètres de la classe de stockage sont :

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

Les attributs de stockage et leurs valeurs possibles peuvent être classés en attributs de sélection de pool de stockage et en attributs Kubernetes.

Attributs de sélection du pool de stockage

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

Attribut	Type	Valeurs	Offre	Demande	Soutenu par
médias[1]	chaîne	hdd, hybride, ssd	Le pool contient des médias de ce type ; hybride signifie les deux	Type de média spécifié	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, solidfire-san

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

[1]: Non pris en charge par ONTAP Select

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

Idéalement, vous pouvez utiliser `attributes` seul pour modéliser les qualités du stockage dont vous avez besoin pour satisfaire les besoins d'une classe particulière. Trident découvre et sélectionne automatiquement les pools de stockage qui correspondent à *tous* les `attributes` que vous spécifiez.

Si vous n'arrivez pas à utiliser `attributes` pour sélectionner automatiquement les bons pools pour une classe, vous pouvez utiliser les paramètres `storagePools` et `additionalStoragePools` pour affiner davantage les pools ou même sélectionner un ensemble spécifique de pools.

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

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

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

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

Dans les `storagePools` et `additionalStoragePools` paramètres, chaque entrée prend la forme `<backend>:<storagePoolList>`, où `<storagePoolList>` est une liste de pools de stockage séparés par des virgules pour le backend spécifié. Par exemple, une valeur pour `additionalStoragePools` pourrait ressembler à `ontapnas_192.168.1.100:aggr1,aggr2;solidfire_192.168.1.101:bronze`. Ces listes acceptent des valeurs regex pour le backend et les valeurs de la liste. Vous pouvez utiliser `tridentctl get backend` pour obtenir la liste des backends et de leurs pools.

Attributs Kubernetes

Ces attributs n'ont aucune incidence sur la sélection des pools de stockage/backends par Trident lors du provisionnement dynamique. Au lieu de cela, ces attributs fournissent simplement des paramètres pris en charge par les volumes persistants Kubernetes. Les nœuds de travail sont responsables des opérations de création du système de fichiers et peuvent nécessiter des utilitaires de système de fichiers, tels que xfsprogs.

Attribut	Type	Valeurs	Description	Pilotes pertinents	Version Kubernetes
fsType	chaîne	ext4, ext3, xfs	Le type de système de fichiers pour les volumes de blocs	solidfire-san, ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, ontap-san-economy	Tous
allowVolumeExpansion	booléen	vrai, faux	Activer ou désactiver la prise en charge de l'agrandissement de la taille du PVC	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, ontap-san-economy, solidfire-san, azure-netapp-files	1.11+
volumeBindingMode	chaîne	Immédiat, WaitForFirstConsumer	Choisissez quand la liaison de volume et le provisionnement dynamique ont lieu	Tous	1.19 - 1.26

ASTUCE

- Le `fsType` paramètre est utilisé pour contrôler le type de système de fichiers souhaité pour les LUN SAN. De plus, Kubernetes utilise également la présence de `fsType` dans une classe de stockage pour indiquer qu'un système de fichiers existe. La propriété du volume peut être contrôlée à l'aide du `fsGroup` contexte de sécurité d'un pod uniquement si `fsType` est défini. Consultez "Kubernetes : configurer un contexte de sécurité pour un pod ou un conteneur" pour un aperçu de la configuration de la propriété du volume à l'aide du `fsGroup` contexte. Kubernetes appliquera la valeur `fsGroup` uniquement si :
  - `fsType` est défini dans la classe de stockage.
  - Le mode d'accès au PVC est RWO.

  Pour les pilotes de stockage NFS, un système de fichiers existe déjà dans l'exportation NFS. Pour utiliser `fsGroup` la classe de stockage, il faut toujours spécifier un `fsType`. Vous pouvez le définir sur `nfs` ou sur toute valeur non nulle.
- Reportez-vous à "Étendre les volumes" pour plus de détails sur l'expansion du volume.
- Le bundle d'installation Trident fournit plusieurs exemples de définitions de classes de stockage à utiliser avec Trident dans `sample-input/storage-class-*.yaml`. La suppression d'une classe de stockage Kubernetes entraîne également la suppression de la classe de stockage Trident correspondante.

Objets `VolumeSnapshotClass` Kubernetes

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

A `VolumeSnapshotClass` doit être défini par un administrateur afin de créer des snapshots. Une classe de snapshot de volume est créée avec la définition suivante :

```
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshotClass
metadata:
  name: csi-snapclass
driver: csi.trident.netapp.io
deletionPolicy: Delete
```

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

Objets `VolumeSnapshot` Kubernetes

Un objet Kubernetes `VolumeSnapshot` est une demande de création d'un instantané d'un volume. De même qu'un PVC représente une demande faite par un utilisateur pour un volume, un instantané de volume est une

demande faite par un utilisateur pour créer un instantané d'un PVC existant.

Lorsqu'une demande de snapshot de volume est reçue, Trident gère automatiquement la création du snapshot pour le volume sur le backend et l'expose en créant un objet `VolumeSnapshotContent` unique. Vous pouvez créer des snapshots à partir de PVC existants et utiliser les snapshots comme DataSource lors de la création de nouveaux PVC.

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 PVC auquel sont associés des snapshots, Trident marque le volume sous-jacent de ce PVC comme étant en **Deleting**, mais ne le supprime pas complètement. Le volume est supprimé lorsque tous les snapshots associés sont supprimés.

Objets `VolumeSnapshotContent` Kubernetes

Un objet Kubernetes `VolumeSnapshotContent` représente un instantané pris à partir d'un volume déjà provisionné. Il est analogue à un `PersistentVolume` et désigne un instantané provisionné sur le cluster de stockage. À l'instar de `PersistentVolumeClaim` et de `PersistentVolume` objets, lorsqu'un instantané est créé, l'objet `VolumeSnapshotContent` maintient une correspondance un-à-un avec l'objet `VolumeSnapshot` qui a demandé la création de l'instantané.

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

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

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

Objets `VolumeGroupSnapshotClass` Kubernetes

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

A `VolumeGroupSnapshotClass` doit être défini par un administrateur afin de créer un groupe d'instantanés. Une classe d'instantané de groupe de volumes est créée avec la définition suivante :

```
apiVersion: groupsnapshot.storage.k8s.io/v1beta1
kind: VolumeGroupSnapshotClass
metadata:
  name: csi-group-snap-class
  annotations:
    kubernetes.io/description: "Trident group snapshot class"
driver: csi.trident.netapp.io
deletionPolicy: Delete
```

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

Objets `VolumeGroupSnapshot` Kubernetes

Un objet Kubernetes `VolumeGroupSnapshot` correspond à une requête de création d'un instantané de plusieurs volumes. De même qu'un PVC représente une requête d'un utilisateur pour un volume, un instantané de groupe de volumes correspond à une requête d'un utilisateur visant à créer un instantané d'un PVC existant.

Lorsqu'une demande de snapshot de groupe de volumes est reçue, Trident gère automatiquement la création du snapshot de groupe pour les volumes sur le backend et expose le snapshot en créant un objet `VolumeGroupSnapshotContent` unique. Vous pouvez créer des snapshots à partir de PVC existants et utiliser les snapshots comme DataSource lors de la création de nouveaux PVC.

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 PVC auquel sont associés des snapshots, Trident marque le volume sous-jacent de ce PVC comme étant en **Deleting**, mais ne le supprime pas complètement. Le snapshot du groupe de volumes est supprimé lorsque tous les snapshots associés sont supprimés.

Objets `VolumeGroupSnapshotContent` Kubernetes

Un objet Kubernetes `VolumeGroupSnapshotContent` représente un instantané de groupe pris à partir d'un volume déjà provisionné. Il est analogue à un `PersistentVolume` et désigne un instantané provisionné sur le cluster de stockage. À l'instar de `PersistentVolumeClaim` et de `PersistentVolume` objets, lorsqu'un instantané est créé, l'objet `VolumeSnapshotContent` maintient une correspondance un-à-un avec l'objet `VolumeSnapshot` qui a demandé la création de l'instantané.

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

Lorsqu'une requête de snapshot est reçue, Trident crée le snapshot du groupe de volumes sur le backend. Après la création du snapshot du groupe de volumes, Trident configure un `VolumeGroupSnapshotContent`

objet et expose ainsi le snapshot à l'API Kubernetes.

Objets `CustomResourceDefinition` Kubernetes

Les ressources personnalisées Kubernetes sont des points de terminaison de l'API Kubernetes définis par l'administrateur et utilisées pour regrouper des objets similaires. Kubernetes prend en charge la création de ressources personnalisées pour stocker une collection d'objets. Vous pouvez obtenir ces définitions de ressources en exécutant `kubectl get crds`.

Les définitions de ressources personnalisées (CRD) et leurs métadonnées d'objet associées sont stockées par Kubernetes dans son système de métadonnées. Cela élimine le besoin d'un système de stockage séparé pour Trident.

Trident utilise `CustomResourceDefinition` des objets pour préserver l'identité des objets Trident, tels que les backends Trident, les classes de stockage Trident et les volumes Trident. Ces objets sont gérés par Trident. De plus, le framework de snapshots de volumes CSI introduit certains CRD nécessaires pour définir les snapshots de volumes.

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

Voici quelques points à retenir concernant les CRD de Trident :

- Lorsque Trident est installé, un ensemble de CRD est créé et peut être utilisé comme n'importe quel autre type de ressource.
- Lors de la désinstallation de Trident à l'aide de la commande `tridentctl uninstall`, les pods Trident sont supprimés, mais les CRD créées ne sont pas nettoyées. Consultez "Désinstaller Trident" pour comprendre comment Trident peut être complètement supprimé et reconfiguré à partir de zéro.

Trident `StorageClass` objets

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

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

Les classes de stockage définissent un ensemble d'exigences pour les volumes. Trident associe ces exigences aux attributs présents dans chaque pool de stockage ; s'ils correspondent, ce pool de stockage est une cible valide pour le provisionnement de volumes utilisant cette classe de stockage.

Vous pouvez créer des configurations de classes de stockage pour définir directement des classes de stockage en utilisant l'API REST. Cependant, pour les déploiements Kubernetes, nous nous attendons à ce qu'elles soient créées lors de l'enregistrement de nouveaux objets `StorageClass` Kubernetes.

Objets backend Trident

Les backends représentent les fournisseurs de stockage sur lesquels Trident provisionne les volumes ; une seule instance de Trident peut gérer un nombre quelconque de backends.

REMARQUE	Il s'agit de l'un des deux types d'objets que vous créez et gérez vous-même. L'autre est l'objet `StorageClass` Kubernetes.

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

Trident `StoragePool` objets

Les pools de stockage représentent les emplacements distincts disponibles pour le provisionnement sur chaque backend. Pour ONTAP, ils correspondent à des agrégats dans les SVM. Pour NetApp HCI/SolidFire, ils correspondent à des niveaux de QoS définis par l'administrateur. Chaque pool de stockage possède un ensemble d'attributs de stockage spécifiques, qui définissent ses caractéristiques de performance et de protection des données.

Contrairement aux autres objets ici, les candidats au pool de stockage sont toujours découverts et gérés automatiquement.

Trident `Volume` objets

Les volumes constituent l'unité de base du provisionnement et comprennent les points de terminaison backend, tels que les partages NFS, ainsi que les LUN iSCSI et FC. Dans Kubernetes, ils correspondent directement à `PersistentVolumes`. Lorsque vous créez un volume, assurez-vous qu'il possède une classe de stockage, qui détermine où ce volume peut être provisionné, ainsi qu'une taille.

REMARQUE	• Dans Kubernetes, ces objets sont gérés automatiquement. Vous pouvez les consulter pour voir ce que Trident a provisionné. • Lors de la suppression d'un PV avec des snapshots associés, le volume Trident correspondant passe à l'état **Deleting**. Pour que le volume Trident soit supprimé, vous devez supprimer les snapshots du volume.

Une configuration de volume définit les propriétés qu'un volume provisionné doit avoir.

Attribut	Type	Obligatoire	Description
version	chaîne	non	Version de l'API Trident (« 1 »)
nom	chaîne	Oui	Nom du volume à créer
storageClass	chaîne	Oui	Classe de stockage à utiliser lors du provisionnement du volume
taille	chaîne	Oui	Taille du volume à provisionner en octets
protocole	chaîne	non	Type de protocole à utiliser : "file" ou "block"
internalName	chaîne	non	Nom de l'objet sur le système de stockage ; généré par Trident

Attribut	Type	Obligatoire	Description
cloneSourceVolume	chaîne	non	ontap (nas, san) & solidfire-*: Nom du volume à cloner
splitOnClone	chaîne	non	ontap (nas, san): Séparer le clone de son parent
snapshotPolicy	chaîne	non	ontap-*: Stratégie d'instantané à utiliser
snapshotReserve	chaîne	non	ontap-*: Pourcentage du volume réservé aux instantanés
exportPolicy	chaîne	non	ontap-nas* : Politique d'export à utiliser
snapshotDirectory	bool	non	ontap-nas* : Whether the snapshot directory is visible
unixPermissions	chaîne	non	ontap-nas*: Permissions UNIX initiales
blockSize	chaîne	non	solidfire-*: Taille du bloc/secteur
fileSystem	chaîne	non	Type de système de fichiers
skipRecoveryQueue	chaîne	non	Lors de la suppression d'un volume, ignorez la file d'attente de récupération dans le stockage et supprimez le volume immédiatement.

Trident génère `internalName` lors de la création du volume. Cela consiste en deux étapes. Tout d'abord, il ajoute le préfixe de stockage (soit le préfixe par défaut `trident` soit le préfixe dans la configuration du backend) au nom du volume, ce qui donne un nom de la forme `<prefix>-<volume-name>`. Il procède ensuite à la normalisation du nom, en remplaçant les caractères non autorisés dans le backend. Pour les backends ONTAP, il remplace les tirets par des traits de soulignement (ainsi, le nom interne devient `<prefix>_<volume-name>`). Pour les backends Element, il remplace les traits de soulignement par des tirets.

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

Trident `Snapshot` objets

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

données du snapshot.

Chaque `Snapshot` objet comprend les propriétés énumérées ci-dessous :

Attribut	Type	Obligatoire	Description
version	Chaîne	Oui	Version de l'API Trident (« 1 »)
nom	Chaîne	Oui	Nom de l'objet instantané Trident
internalName	Chaîne	Oui	Nom de l'objet snapshot Trident sur le système de stockage
volumeName	Chaîne	Oui	Nom du volume persistant pour lequel l'instantané est créé
volumeInternalName	Chaîne	Oui	Nom de l'objet volume Trident associé sur le système de stockage

REMARQUE Dans Kubernetes, ces objets sont gérés automatiquement. Vous pouvez les consulter pour voir ce que Trident a provisionné.

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

Trident `ResourceQuota` objet

Le daemonset Trident consomme une `system-node-critical`Priority Class — la Priority Class la plus élevée disponible dans Kubernetes — pour garantir que Trident puisse identifier et nettoyer les volumes lors de l'arrêt progressif des nœuds et permettre aux pods du daemonset Trident de préempter les charges de travail de priorité inférieure dans les clusters où la pression sur les ressources est élevée.

Pour ce faire, Trident utilise un `ResourceQuota` objet afin de garantir que la classe de priorité « system-node-critical » sur le daemonset Trident soit respectée. Avant le déploiement et la création du daemonset, Trident recherche l' `ResourceQuota` objet et, s'il n'est pas trouvé, l'applique.

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

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

```
apiVersion: <version>
kind: ResourceQuota
metadata:
  name: trident-csi
  labels:
    app: node.csi.trident.netapp.io
spec:
  scopeSelector:
    matchExpressions:
      - operator: In
        scopeName: PriorityClass
        values:
          - system-node-critical
```

Pour plus d'informations sur les quotas de ressources, veuillez consulter "Kubernetes : Quotas de ressources".

Nettoyez `ResourceQuota` si l'installation échoue

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

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

Retirer `ResourceQuota`

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

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

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

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

REMARQUE	PSS remplace les Pod Security Policies (PSP). PSP a été déprécié dans Kubernetes v1.21 et sera supprimé dans la v1.25. Pour plus d'informations, consultez "Kubernetes : sécurité".

Contexte de sécurité Kubernetes requis et champs associés

Autorisation	Description
Privilégié	CSI exige que les points de montage soient bidirectionnels, ce qui signifie que le pod Trident du nœud doit exécuter un conteneur privilégié. Pour plus d'informations, consultez "Kubernetes : propagation du montage".
Réseau hôte	Nécessaire pour le démon iSCSI. `iscsiadm` Gère les montages iSCSI et utilise le réseau hôte pour communiquer avec le démon iSCSI.
IPC de l'hôte	NFS utilise la communication interprocessus (IPC) pour communiquer avec le NFSD.
PID de l'hôte	Nécessaire pour démarrer `rpc-statd` pour NFS. Trident interroge les processus hôtes pour déterminer si `rpc-statd` est en cours d'exécution avant de monter les volumes NFS.
Capacités	La `SYS_ADMIN` fonctionnalité est fournie dans le cadre des fonctionnalités par défaut pour les conteneurs privilégiés. Par exemple, Docker définit ces fonctionnalités pour les conteneurs privilégiés : `CapPrm: 0000003fffffffff` `CapEff: 0000003fffffffff`
Seccomp	Le profil Seccomp est toujours « non confiné » dans les conteneurs privilégiés ; par conséquent, il ne peut pas être activé dans Trident.
SELinux	Sur OpenShift, les conteneurs privilégiés s'exécutent dans le domaine `spc_t` (« Super Privileged Container »), et les conteneurs non privilégiés s'exécutent dans le domaine `container_t`. Sur `containerd`, avec `container-selinux` installé, tous les conteneurs s'exécutent dans le domaine `spc_t`, ce qui désactive effectivement SELinux. Par conséquent, Trident n'ajoute pas `seLinuxOptions` aux conteneurs.
DAC	Les conteneurs privilégiés doivent être exécutés en tant que root. Les conteneurs non privilégiés s'exécutent en tant que root pour accéder aux sockets unix requis par CSI.

Normes de sécurité des pods (PSS)

Étiquette	Description	Défaut
`pod-security.kubernetes.io/enforce` `pod-security.kubernetes.io/enforce-version`	Autorise le contrôleur Trident et les nœuds à être admis dans l'espace de noms d'installation. Ne modifiez pas l'étiquette de l'espace de noms.	`enforce: privileged` `enforce-version: <version of the current cluster or highest version of PSS tested.>`

AVERTISSEMENT	Modifier les étiquettes d'espace de noms peut empêcher la planification des pods, entraîner une « Error creating: … » ou un « Warning: trident-csi-… ». Si cela se produit, vérifiez si l'étiquette d'espace de noms pour `privileged` a été modifiée. Si c'est le cas, réinstallez Trident.

Politiques de sécurité des pods (PSP)

Champ	Description	Défaut
`allowPrivilegeEscalation`	Les conteneurs privilégiés doivent autoriser l'élévation de privilèges.	`true`
`allowedCSIDrivers`	Trident n'utilise pas de volumes CSI éphémères en ligne.	Vide
`allowedCapabilities`	Les conteneurs Trident non privilégiés ne nécessitent pas plus de capacités que l'ensemble par défaut et les conteneurs privilégiés se voient accorder toutes les capacités possibles.	Vide
`allowedFlexVolumes`	Trident n'utilise pas de "Pilote FlexVolume", par conséquent, ils ne sont pas inclus dans la liste des volumes autorisés.	Vide
`allowedHostPaths`	Le pod Trident du nœud monte le système de fichiers racine du nœud, par conséquent, la configuration de cette liste ne présente aucun avantage.	Vide
`allowedProcMountTypes`	Trident n'utilise aucun `ProcMountTypes`.	Vide
`allowedUnsafeSysctls`	Trident ne nécessite aucune opération non sécurisée `sysctls`.	Vide
`defaultAddCapabilities`	Aucune capacité n'est requise pour être ajoutée aux conteneurs privilégiés.	Vide
`defaultAllowPrivilegeEscalation`	L'autorisation d'élévation de privilèges est gérée dans chaque pod Trident.	`false`
`forbiddenSysctls`	Aucun `sysctls` n'est autorisé.	Vide
`fsGroup`	Les conteneurs Trident s'exécutent en tant que root.	`RunAsAny`
`hostIPC`	Le montage de volumes NFS nécessite que l'IPC de l'hôte communique avec `nfsd`	`true`

Champ	Description	Défaut
`hostNetwork`	iscsiadm nécessite que le réseau hôte communique avec le démon iSCSI.	`true`
`hostPID`	L'identifiant du processus hôte (PID) est requis pour vérifier si `rpc-statd` est en cours d'exécution sur le nœud.	`true`
`hostPorts`	Trident n'utilise aucun port hôte.	Vide
`privileged`	Les pods de nœuds Trident doivent exécuter un conteneur privilégié pour pouvoir monter des volumes.	`true`
`readOnlyRootFilesystem`	Les pods de nœud Trident doivent écrire sur le système de fichiers du nœud.	`false`
`requiredDropCapabilities`	Les pods de nœuds Trident exécutent un conteneur privilégié et ne peuvent pas supprimer de fonctionnalités.	`none`
`runAsGroup`	Les conteneurs Trident s'exécutent en tant que root.	`RunAsAny`
`runAsUser`	Les conteneurs Trident s'exécutent en tant que root.	`runAsAny`
`runtimeClass`	Trident n'utilise pas `RuntimeClasses`.	Vide
`seLinux`	Trident ne configure pas `seLinuxOptions`car il existe actuellement des différences dans la manière dont les environnements d'exécution de conteneurs et les distributions Kubernetes gèrent SELinux.	Vide
`supplementalGroups`	Les conteneurs Trident s'exécutent en tant que root.	`RunAsAny`
`volumes`	Les pods Trident nécessitent ces plugins de volume.	`hostPath, projected, emptyDir`

Security Context Constraints (SCC)

Étiquettes	Description	Défaut
`allowHostDirVolumePlugin`	Les pods de nœud Trident montent le système de fichiers racine du nœud.	`true`

Étiquettes	Description	Défaut
`allowHostIPC`	Le montage des volumes NFS nécessite que l'IPC de l'hôte communique avec `nfsd`.	`true`
`allowHostNetwork`	iscsiadm nécessite que le réseau hôte communique avec le démon iSCSI.	`true`
`allowHostPID`	L'identifiant du processus hôte (PID) est requis pour vérifier si `rpc-statd` est en cours d'exécution sur le nœud.	`true`
`allowHostPorts`	Trident n'utilise aucun port hôte.	`false`
`allowPrivilegeEscalation`	Les conteneurs privilégiés doivent autoriser l'élévation de privilèges.	`true`
`allowPrivilegedContainer`	Les pods de nœuds Trident doivent exécuter un conteneur privilégié pour pouvoir monter des volumes.	`true`
`allowedUnsafeSysctls`	Trident ne nécessite aucune opération non sécurisée `sysctls`.	`none`
`allowedCapabilities`	Les conteneurs Trident non privilégiés ne nécessitent pas plus de capacités que l'ensemble par défaut et les conteneurs privilégiés se voient accorder toutes les capacités possibles.	Vide
`defaultAddCapabilities`	Aucune capacité n'est requise pour être ajoutée aux conteneurs privilégiés.	Vide
`fsGroup`	Les conteneurs Trident s'exécutent en tant que root.	`RunAsAny`
`groups`	Ce SCC est spécifique à Trident et est lié à son utilisateur.	Vide
`readOnlyRootFilesystem`	Les pods de nœud Trident doivent écrire sur le système de fichiers du nœud.	`false`
`requiredDropCapabilities`	Les pods de nœuds Trident exécutent un conteneur privilégié et ne peuvent pas supprimer de fonctionnalités.	`none`
`runAsUser`	Les conteneurs Trident s'exécutent en tant que root.	`RunAsAny`

Étiquettes	Description	Défaut
`seLinuxContext`	Trident ne configure pas `seLinuxOptions`car il existe actuellement des différences dans la manière dont les environnements d'exécution de conteneurs et les distributions Kubernetes gèrent SELinux.	Vide
`seccompProfiles`	Les conteneurs privilégiés s'exécutent toujours en mode "Unconfined".	Vide
`supplementalGroups`	Les conteneurs Trident s'exécutent en tant que root.	`RunAsAny`
`users`	Une entrée est prévue pour lier ce SCC à l'utilisateur Trident dans l'espace de noms Trident.	s.o.
`volumes`	Les pods Trident nécessitent ces plugins de volume.	`hostPath, downwardAPI, projected, emptyDir`

Mentions légales

Les mentions légales donnent accès aux déclarations de droits d'auteur, aux marques de commerce, aux brevets et plus encore.

Copyright

"https://www.netapp.com/company/legal/copyright/"

Marques déposées

NETAPP, le logo NETAPP et les marques figurant sur la page NetApp Trademarks sont des marques de NetApp, Inc. Les autres noms de sociétés et de produits peuvent être des marques de leurs propriétaires respectifs.

"https://www.netapp.com/company/legal/trademarks/"

Brevets

Une liste actuelle des brevets détenus par NetApp peut être trouvée à :

https://www.netapp.com/pdf.html?item=/media/11887-patentspage.pdf

Politique de confidentialité

"https://www.netapp.com/company/legal/privacy-policy/"

Open source

Vous pouvez consulter les droits d'auteur et les licences de tiers utilisés dans NetApp software for Trident dans le fichier de notifications de chaque version à 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.