Commencez

Astra Trident

NetApp
April 04, 2024

Sommaire

Commencez

Essayez-le

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

Découvrez Test Drive

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

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

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

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

De formation

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

Découvrez Astra Trident 23.01, un document essentiel

Vous devez lire les renseignements essentiels suivants sur Astra Trident.

** informations pratiques sur le Tridécouvrez Astra **

- Kubernetes 1.27 est désormais pris en charge dans Trident. Mise à niveau de Trident avant la mise à niveau de Kubernetes.
- Astra Trident applique rigoureusement l'utilisation de la configuration de chemins d'accès multiples dans les environnements SAN, avec la valeur recommandée de `find_multipaths: no` dans le fichier multipath.conf.

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

Systèmes front-end (orchestrateurs) pris en charge

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

- Anthos sur site (VMware) et Anthos sur bare Metal 1.12
- Kubernetes 1.21 - 1.27
- Moteur Kubernetes Mirantis 3.5
- OpenShift 4.9 - 4.12

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

- Anthos sur site (VMware) et Anthos sur bare Metal 1.12
- Kubernetes 1.21 - 1.27
- OpenShift 4.9 - 4.12

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

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

Systèmes back-end pris en charge (stockage)

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

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

Configuration requise

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

Fonction	Version Kubernetes	Portes-fonctions requises ?
CSI Trident	1.21 - 1.27	Non
Snapshots de volume	1.21 - 1.27	Non
Volume persistant à partir des copies Snapshot des volumes	1.21 - 1.27	Non
Redimensionnement PV iSCSI	1.21 - 1.27	Non

Fonction	Version Kubernetes	Portes-fonctions requises ?
Chap bidirectionnel ONTAP	1.21 - 1.27	Non
Règles d'exportation dynamiques	1.21 - 1.27	Non
Opérateur Trident	1.21 - 1.27	Non
Topologie CSI	1.21 - 1.27	Non

Systèmes d'exploitation hôtes testés

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

- Versions de Red Hat CoreOS (RHCOS) prises en charge par OpenShift Container Platform (AMD64 et ARM64)
- RHEL 8+ (AMD64 ET ARM64)
- Ubuntu 22.04 ou version ultérieure (AMD64 et ARM64)
- Windows Server 2019 (AMD64)

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

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

Configuration de l'hôte

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

"Préparez le nœud de travail"

Configuration du système de stockage

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

"Configuration des systèmes back-end"

Ports Trident d'Astra

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

"Ports Trident d'Astra"

Images de conteneur et versions Kubernetes correspondantes

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

Version Kubernetes	Image de conteneur
v1.21.0	• docker.io/netapp/trident : 23.04.0 • docker.io/netapp/trident-autosupport:23.04 • registry.k8s.io/sig-storage/csi-provisionneur:v3.4.1 • registry.k8s.io/sig-storage/csi-attacher:v4.2.0 • registry.k8s.io/sig-storage/csi-resizer:v1.7.0 • registry.k8s.io/sig-storage/csi-snapshotter:v6.2.1 • registry.k8s.io/sig-storage/csi-node-driver-registratr:v2.7.0 • docker.io/netapp/trident-operator:23.04.0 (en option)
v1.22.0	• docker.io/netapp/trident : 23.04.0 • docker.io/netapp/trident-autosupport:23.04 • registry.k8s.io/sig-storage/csi-provisionneur:v3.4.1 • registry.k8s.io/sig-storage/csi-attacher:v4.2.0 • registry.k8s.io/sig-storage/csi-resizer:v1.7.0 • registry.k8s.io/sig-storage/csi-snapshotter:v6.2.1 • registry.k8s.io/sig-storage/csi-node-driver-registratr:v2.7.0 • docker.io/netapp/trident-operator:23.04.0 (en option)
v1.23.0	• docker.io/netapp/trident : 23.04.0 • docker.io/netapp/trident-autosupport:23.04 • registry.k8s.io/sig-storage/csi-provisionneur:v3.4.1 • registry.k8s.io/sig-storage/csi-attacher:v4.2.0 • registry.k8s.io/sig-storage/csi-resizer:v1.7.0 • registry.k8s.io/sig-storage/csi-snapshotter:v6.2.1 • registry.k8s.io/sig-storage/csi-node-driver-registratr:v2.7.0 • docker.io/netapp/trident-operator:23.04.0 (en option)

Version Kubernetes	Image de conteneur
v1.24.0	• docker.io/netapp/trident : 23.04.0 • docker.io/netapp/trident-autosupport:23.04 • registry.k8s.io/sig-storage/csi-provisionneur:v3.4.1 • registry.k8s.io/sig-storage/csi-attacher:v4.2.0 • registry.k8s.io/sig-storage/csi-resizer:v1.7.0 • registry.k8s.io/sig-storage/csi-snapshotter:v6.2.1 • registry.k8s.io/sig-storage/csi-node-driver-registratr:v2.7.0 • docker.io/netapp/trident-operator:23.04.0 (en option)
v1.25.0	• docker.io/netapp/trident : 23.04.0 • docker.io/netapp/trident-autosupport:23.04 • registry.k8s.io/sig-storage/csi-provisionneur:v3.4.1 • registry.k8s.io/sig-storage/csi-attacher:v4.2.0 • registry.k8s.io/sig-storage/csi-resizer:v1.7.0 • registry.k8s.io/sig-storage/csi-snapshotter:v6.2.1 • registry.k8s.io/sig-storage/csi-node-driver-registratr:v2.7.0 • docker.io/netapp/trident-operator:23.04.0 (en option)
v1.26.0	• docker.io/netapp/trident : 23.04.0 • docker.io/netapp/trident-autosupport:23.04 • registry.k8s.io/sig-storage/csi-provisionneur:v3.4.1 • registry.k8s.io/sig-storage/csi-attacher:v4.2.0 • registry.k8s.io/sig-storage/csi-resizer:v1.7.0 • registry.k8s.io/sig-storage/csi-snapshotter:v6.2.1 • registry.k8s.io/sig-storage/csi-node-driver-registratr:v2.7.0 • docker.io/netapp/trident-operator:23.04.0 (en option)

Version Kubernetes	Image de conteneur
v1.27.0	• docker.io/netapp/trident : 23.04.0 • docker.io/netapp/trident-autosupport:23.04 • registry.k8s.io/sig-storage/csi-provisionneur:v3.4.1 • registry.k8s.io/sig-storage/csi-attacher:v4.2.0 • registry.k8s.io/sig-storage/csi-resizer:v1.7.0 • registry.k8s.io/sig-storage/csi-snapshotter:v6.2.1 • registry.k8s.io/sig-storage/csi-node-driver-registratr:v2.7.0 • docker.io/netapp/trident-operator:23.04.0 (en option)

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

Installer Astra Trident

Découvrez l'installation d'Astra Trident

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

Informations stratégiques sur Astra Trident 23.04

Vous devez lire les renseignements essentiels suivants sur Astra Trident.

informations pratiques sur le Tridécouvrez Astra

- Kubernetes 1.27 est désormais pris en charge dans Trident. Mise à niveau de Trident avant la mise à niveau de Kubernetes.
- Astra Trident applique rigoureusement l'utilisation de la configuration de chemins d'accès multiples dans les environnements SAN, avec la valeur recommandée de `find_multipaths: no` dans le fichier multipath.conf.

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

Avant de commencer

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

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

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

Choisissez votre méthode d'installation

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

Utilisation de l'opérateur Trident

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

L'utilisateur de Trident présente les avantages suivants :

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

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

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

** - Capcuratif de la prise **

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

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

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

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

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

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

** : mise à niveau de **

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

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

Gestion de clusters ™ avec BlueXP (anciennement Cloud Manager)

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

À l'aide de `tridentctl`

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

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

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

Choisissez votre mode d’installation

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

Installation standard

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

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

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

Installation hors ligne

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

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

Installation à distance

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

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

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

Après avoir pris vos décisions, sélectionnez le processus approprié.

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

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

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

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

Autres options de configuration connues

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

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

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

Installer à l'aide de l'opérateur Trident

Déployer manuellement l'opérateur Trident (mode Standard)

Vous pouvez déployer manuellement l'opérateur Trident pour installer Astra Trident. Ce processus s'applique aux installations dans lesquelles les images de conteneur requises

par Astra Trident ne sont pas stockées dans un registre privé. Si vous disposez d'un registre d'images privé, utilisez le "processus de déploiement hors ligne".

Informations stratégiques sur Astra Trident 23.04

Vous devez lire les renseignements essentiels suivants sur Astra Trident.

** informations pratiques sur le Tridécouvrez Astra **

- Kubernetes 1.27 est désormais pris en charge dans Trident. Mise à niveau de Trident avant la mise à niveau de Kubernetes.
- Astra Trident applique rigoureusement l'utilisation de la configuration de chemins d'accès multiples dans les environnements SAN, avec la valeur recommandée de `find_multipaths: no` dans le fichier multipath.conf.

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

Déployer manuellement l'opérateur Trident et installer Trident

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

Avant de commencer

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

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

1. Vérifiez votre version Kubernetes :

```
kubectl version
```

2. Vérifiez les privilèges d'administrateur du cluster :

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

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

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

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

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

```
wget https://github.com/NetApp/trident/releases/download/v23.04.0/trident-
installer-23.04.0.tar.gz
tar -xf trident-installer-23.04.0.tar.gz
cd trident-installer
```

Étape 2 : créez le `TridentOrchestrator` CRD

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

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

Étape 3 : déployer l'opérateur Trident

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

- Pour les clusters exécutant Kubernetes 1.24 ou version antérieure, utilisez `bundle_pre_1_25.yaml`.

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

Avant de commencer

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

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

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

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

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

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

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

Étapes

1. Créer les ressources et déployer l'opérateur :

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

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

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

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

Étape 4 : créez le `TridentOrchestrator` Et installer Trident

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

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

kubectl describe torc trident

Name:        trident
Namespace:
Labels:      <none>
Annotations: <none>
API Version: trident.netapp.io/v1
Kind:        TridentOrchestrator
...
Spec:
  Debug:     true
  Namespace: trident
Status:
  Current Installation Params:
    IPv6:                      false
    Autosupport Hostname:
    Autosupport Image:         netapp/trident-autosupport:23.04
    Autosupport Proxy:
    Autosupport Serial Number:
    Debug:                     true
    Image Pull Secrets:
    Image Registry:
    k8sTimeout:           30
    Kubelet Dir:          /var/lib/kubelet
    Log Format:           text
    Silence Autosupport:  false
    Trident Image:        netapp/trident:23.04.0
  Message:                  Trident installed  Namespace:
trident
  Status:                   Installed
  Version:                  v23.04.0
Events:
    Type Reason Age From Message ---- ------ ---- ---- -------Normal
    Installing 74s trident-operator.netapp.io Installing Trident Normal
    Installed 67s trident-operator.netapp.io Trident installed
```

Vérifiez l'installation

Il existe plusieurs façons de vérifier votre installation.

À l'aide de `TridentOrchestrator` état

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

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

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

Utilisation du statut de création du pod

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

```
kubectl get pods -n trident

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

À l'aide de `tridentctl`

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

```
./tridentctl -n trident version

+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 23.04.0        | 23.04.0        |
+----------------+----------------+
```

Et la suite

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

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

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

Informations stratégiques sur Astra Trident 23.04

Vous devez lire les renseignements essentiels suivants sur Astra Trident.

informations pratiques sur le Tridécouvrez Astra

- Kubernetes 1.27 est désormais pris en charge dans Trident. Mise à niveau de Trident avant la mise à niveau de Kubernetes.
- Astra Trident applique rigoureusement l'utilisation de la configuration de chemins d'accès multiples dans les environnements SAN, avec la valeur recommandée de `find_multipaths: no` dans le fichier multipath.conf.

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

Déployer manuellement l'opérateur Trident et installer Trident

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

Avant de commencer

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

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

1. Vérifiez votre version Kubernetes :

```
kubectl version
```

2. Vérifiez les privilèges d'administrateur du cluster :

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

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

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

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

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

```
wget https://github.com/NetApp/trident/releases/download/v23.04.0/trident-
installer-23.04.0.tar.gz
tar -xf trident-installer-23.04.0.tar.gz
cd trident-installer
```

Étape 2 : créez le `TridentOrchestrator` CRD

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

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

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

Dans `/deploy/operator.yaml`, mettre à jour `image: docker.io/netapp/trident-operator:23.04.0` pour refléter l'emplacement de votre registre d'images. Votre "Images Trident et CSI"

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

- `image: <your-registry>/trident-operator:23.04.0` si vos images sont toutes situées dans un même registre.
- `image: <your-registry>/netapp/trident-operator:23.04.0` Si votre image Trident se trouve dans un registre différent de vos images CSI.

Étape 4 : déploiement de l'opérateur Trident

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

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

Avant de commencer

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

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

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

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

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

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

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

Étapes

1. Créer les ressources et déployer l'opérateur :

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

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

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

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

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

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

Images dans un registre

```
imageRegistry: "<your-registry>"
autosupportImage: "<your-registry>/trident-autosupport:23.04"
tridentImage: "<your-registry>/trident:23.04.0"
```

Images dans différents registres

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

```
imageRegistry: "<your-registry>/sig-storage"
autosupportImage: "<your-registry>/netapp/trident-autosupport:23.04"
tridentImage: "<your-registry>/netapp/trident:23.04.0"
```

Étape 6 : créez le `TridentOrchestrator` Et installer Trident

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

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

kubectl describe torc trident

Name:        trident
Namespace:
Labels:      <none>
Annotations: <none>
API Version: trident.netapp.io/v1
Kind:        TridentOrchestrator
...
Spec:
  Autosupport Image:  <your-registry>/netapp/trident-autosupport:23.04
  Debug:              true
  Image Registry:     <your-registry>/sig-storage
  Namespace:          trident
  Trident Image:      <your-registry>/netapp/trident:23.04.0
Status:
  Current Installation Params:
    IPv6:                       false
    Autosupport Hostname:
    Autosupport Image:          <your-registry>/netapp/trident-
autosupport:23.04
    Autosupport Proxy:
    Autosupport Serial Number:
    Debug:                      true
    Http Request Timeout:       90s
    Image Pull Secrets:
    Image Registry:       <your-registry>/sig-storage
    k8sTimeout:           30
    Kubelet Dir:          /var/lib/kubelet
    Log Format:           text
    Probe Port:           17546
    Silence Autosupport:  false
    Trident Image:        <your-registry>/netapp/trident:23.04.0
  Message:                Trident installed
  Namespace:              trident
  Status:                 Installed
  Version:                v23.04.0
Events:
    Type Reason Age From Message ---- ------ ---- ---- -------Normal
    Installing 74s trident-operator.netapp.io Installing Trident Normal
    Installed 67s trident-operator.netapp.io Trident installed
```

Vérifiez l'installation

Il existe plusieurs façons de vérifier votre installation.

À l'aide de `TridentOrchestrator` état

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

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

Utilisation du statut de création du pod

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

```
kubectl get pods -n trident

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

À l'aide de `tridentctl`

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

```
./tridentctl -n trident version

+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 23.04.0        | 23.04.0        |
+----------------+----------------+
```

Et la suite

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

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

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

Informations stratégiques sur Astra Trident 23.04

Vous devez lire les renseignements essentiels suivants sur Astra Trident.

** informations pratiques sur le Tridécouvrez Astra **

- Kubernetes 1.27 est désormais pris en charge dans Trident. Mise à niveau de Trident avant la mise à niveau de Kubernetes.
- Astra Trident applique rigoureusement l'utilisation de la configuration de chemins d'accès multiples dans les environnements SAN, avec la valeur recommandée de `find_multipaths: no` dans le fichier multipath.conf.

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

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

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

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

Avant de commencer

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

Étapes

1. Ajout du référentiel Astra Trident Helm :

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

2. Utiliser `helm install` et spécifiez un nom pour votre déploiement comme dans l'exemple suivant où `23.04.0` Est la version d'Astra Trident que vous installez.

```
helm install <name> netapp-trident/trident-operator --version 23.04.0
--create-namespace --namespace <trident-namespace>
```

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

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

Transmettre les données de configuration pendant l'installation

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

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

Par exemple, pour modifier la valeur par défaut de `debug`, exécutez ce qui suit `--set` commande où `23.04.0` Est la version d'Astra Trident que vous installez :

```
helm install <name> netapp-trident/trident-operator --version 23.04.0
--create-namespace --namespace --set tridentDebug=true
```

Options de configuration

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

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

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

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

Présentation des pods de contrôleur et des nœuds

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

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

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

Et la suite

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

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

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

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

Informations stratégiques sur Astra Trident 23.04

Vous devez lire les renseignements essentiels suivants sur Astra Trident.

** informations pratiques sur le Tridécouvrez Astra **

- Kubernetes 1.27 est désormais pris en charge dans Trident. Mise à niveau de Trident avant la mise à niveau de Kubernetes.
- Astra Trident applique rigoureusement l'utilisation de la configuration de chemins d'accès multiples dans les environnements SAN, avec la valeur recommandée de `find_multipaths: no` dans le fichier multipath.conf.

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

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

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

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

Avant de commencer

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

Étapes

1. Ajout du référentiel Astra Trident Helm :

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

2. Utiliser `helm install` et spécifiez un nom pour votre déploiement et l'emplacement du registre d'images. Votre "Images Trident et CSI" Peut être situé dans un registre ou dans des registres différents, mais toutes les images CSI doivent se trouver dans le même registre. Dans les exemples, `23.04.0` Est la version d'Astra Trident que vous installez.

Images dans un registre

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

Images dans différents registres

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

```
helm install <name> netapp-trident/trident-operator --version
23.04.0 --set imageRegistry=<your-registry>/sig-storage --set
operatorImage=<your-registry>/netapp/trident-operator:23.04.0 --set
tridentAutosupportImage=<your-registry>/netapp/trident-
autosupport:23.04 --set tridentImage=<your-
registry>/netapp/trident:23.04.0 --create-namespace --namespace
<trident-namespace>
```

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

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

Transmettre les données de configuration pendant l'installation

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

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

Par exemple, pour modifier la valeur par défaut de `debug`, exécutez ce qui suit `--set` commande où `23.04.0` Est la version d'Astra Trident que vous installez :

```
helm install <name> netapp-trident/trident-operator --version 23.04.0
--create-namespace --namespace --set tridentDebug=true
```

Options de configuration

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

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

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

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

Présentation des pods de contrôleur et des nœuds

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

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

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

Et la suite

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

Personnalisez l'installation de l'opérateur Trident

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

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

Présentation des pods de contrôleur et des nœuds

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

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

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

Options de configuration

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

Ce tableau est plus détaillé `TridentOrchestrator` attributs.

Paramètre	Description	Valeur par défaut
`namespace`	Espace de noms pour installer Astra Trident dans	« par défaut »
`debug`	Activez le débogage pour Astra Trident	faux
`enableForceDetach`	`ontap-san` et `ontap-san-economy` uniquement. Fonctionne avec Kubernetes non-Grass Node Shutdown (NGN) pour autoriser les administrateurs du cluster à migrer en toute sécurité les workloads avec des volumes montés vers de nouveaux nœuds en cas de problème. **Il s'agit d'une caractéristique expérimentale dans 23.04.** voir Détails sur le détachement forcé pour plus de détails.	`false`

Paramètre	Description	Valeur par défaut
`windows`	Réglage sur `true` Active l'installation sur les nœuds de travail Windows.	faux
`useIPv6`	Installez Astra Trident sur IPv6	faux
`k8sTimeout`	Délai d'expiration pour les opérations Kubernetes	30 secondes
`silenceAutosupport`	N'envoyez pas de packs AutoSupport à NetApp automatiquement	faux
`autosupportImage`	Image conteneur pour la télémétrie AutoSupport	netapp/trident-autosupport:23.07
`autosupportProxy`	Adresse/port d'un proxy pour l'envoi de AutoSupport Télémétrie	"http://proxy.example.com:8888"
`uninstall`	Indicateur utilisé pour désinstaller Astra Trident	faux
`logFormat`	Format de connexion Astra Trident à utiliser [text,json]	« texte »
`tridentImage`	Image Astra Trident à installer	netapp/trident : 23.07
`imageRegistry`	Chemin d'accès au registre interne, du format `<registry FQDN>[:port][/subpath]`	« k8s.gcr.io/sig-storage » (k8s 1.19+) ou « quay.io/k8scsi »
`kubeletDir`	Chemin d'accès au répertoire kubelet de l'hôte	"/var/lib/kubelet"
`wipeout`	Liste des ressources à supprimer pour effectuer une suppression complète de Astra Trident	
`imagePullSecrets`	Secrets pour extraire des images d'un registre interne	
`imagePullPolicy`	Définit la stratégie de collecte d'image pour l'opérateur Trident. Les valeurs valides sont : `Always` pour toujours tirer l'image. `IfNotPresent` pour extraire l'image uniquement s'il n'existe pas déjà sur le nœud. `Never` pour ne jamais tirer l'image.	`IfNotPresent`

Paramètre	Description	Valeur par défaut
`controllerPluginNodeSelector`	Sélecteurs de nœuds supplémentaires pour les pods. Suit le même format que `pod.spec.nodeSelector`.	Pas de valeur par défaut ; facultatif
`controllerPluginTolerations`	Remplace les tolérances Kubernetes pour les pods. Suit le même format que `pod.spec.Tolerations`.	Pas de valeur par défaut ; facultatif
`nodePluginNodeSelector`	Sélecteurs de nœuds supplémentaires pour les pods. Suit le même format que `pod.spec.nodeSelector`.	Pas de valeur par défaut ; facultatif
`nodePluginTolerations`	Remplace les tolérances Kubernetes pour les pods. Suit le même format que `pod.spec.Tolerations`.	Pas de valeur par défaut ; facultatif

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

Détails sur le détachement forcé

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

Comme Astra Trident repose sur LES NGN Kubernetes, ne supprimez pas `out-of-service` elle est corrompue jusqu'à ce que toutes les charges de travail non tolérables soient replanifiées. L'application ou la suppression imprudemment de cet outil peut compromettre la protection des données back-end.

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

1. Cessez l'accès aux E/S back-end pour les volumes montés sur ce nœud.
2. Marquez l'objet de nœud Astra Trident en tant que `dirty` (pas sûr pour les nouvelles publications).

Le contrôleur Trident rejette les nouvelles demandes de volume publiées jusqu'à ce que le nœud soit de nouveau qualifié (après avoir été marqué comme `dirty`) Par le pod du nœud Trident. Tous les workloads planifiés avec une demande de volume persistant montée (même lorsque le nœud du cluster est sain et prêt) ne seront pas acceptés tant qu'Astra Trident ne peut pas vérifier le nœud `clean` (sûr pour les nouvelles publications).

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

1. Identifiez et nettoyez les chemins publiés obsolètes sur le nœud.

2. Si le nœud est dans un `cleanable` state (le taint hors service a été supprimé et le nœud est dans `Ready` État). Tous les chemins obsolètes et publiés sont propres. Astra Trident reprépare le nœud en tant que `clean` et autoriser les nouveaux volumes publiés sur le nœud.

Exemples de configurations

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

Exemple 1 : configuration personnalisée de base

Voici un exemple de configuration personnalisée de base.

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

Exemple 2 : déploiement avec des sélecteurs de nœuds

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

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

Exemple 3 : déploiement sur des nœuds de travail Windows

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

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

Installation à l'aide de tridentctl

Installation à l'aide de tridentctl

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

Informations stratégiques sur Astra Trident 23.04

Vous devez lire les renseignements essentiels suivants sur Astra Trident.

** informations pratiques sur le Tridécouvrez Astra **

- Kubernetes 1.27 est désormais pris en charge dans Trident. Mise à niveau de Trident avant la mise à niveau de Kubernetes.
- Astra Trident applique rigoureusement l'utilisation de la configuration de chemins d'accès multiples dans les environnements SAN, avec la valeur recommandée de `find_multipaths: no` dans le fichier multipath.conf.

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

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

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

Avant de commencer

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

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

1. Vérifiez votre version Kubernetes :

```
kubectl version
```

2. Vérifiez les privilèges d'administrateur du cluster :

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

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

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

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

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

```
wget https://github.com/NetApp/trident/releases/download/v23.04.0/trident-
installer-23.04.0.tar.gz
tar -xf trident-installer-23.04.0.tar.gz
cd trident-installer
```

Étape 2 : installez Astra Trident

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

Mode standard

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

Images dans un registre

```
./tridentctl install -n trident --image-registry <your-registry>
--autosupport-image <your-registry>/trident-autosupport:23.04 --trident
-image <your-registry>/trident:23.04.0
```

Images dans différents registres

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

```
./tridentctl install -n trident --image-registry <your-registry>/sig-
storage --autosupport-image <your-registry>/netapp/trident-
autosupport:23.04 --trident-image <your-
registry>/netapp/trident:23.04.0
```

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=23.04.0
INFO Trident installation succeeded.
....
```

Vérifiez l'installation

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

Utilisation du statut de création du pod

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

```
kubectl get pods -n trident

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

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

À l'aide de `tridentctl`

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

```
./tridentctl -n trident version

+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 23.04.0        | 23.04.0        |
+----------------+----------------+
```

Exemples de configurations

Exemple 1 : activez Astra Trident pour qu'elle s'exécute sur les nœuds Windows

Pour permettre l'exécution d'Astra Trident sur les nœuds Windows :

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

Exemple 2 : activer le détachement forcé

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

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

Et la suite

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

Personnalisez l'installation tridentctl

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

En savoir plus sur le programme d'installation

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

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

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

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

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

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

Et la suite ?

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

Étape 1 : créer un back-end

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

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

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

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

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

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

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

Étape 2 : créer une classe de stockage

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

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

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

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

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

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

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

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

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

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

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

Étape 3 : provisionner le premier volume

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

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

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

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

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

Étape 4 : montez les volumes dans un pod

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

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

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

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

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

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

Pour supprimer le volume, supprimez la réclamation :

```
kubectl delete pvc basic
```

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

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

Informations sur le copyright

Informations sur les marques commerciales