# Commencez

## Astra Trident

NetApp
April 16, 2024

# Sommaire

# Commencez

## Essayez-le

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

Lancez-vous en examinant les systèmes front-end, les systèmes back-end et la configuration hôte pris en charge.

> Pour en savoir plus sur les ports qu'utilise Astra Trident, consultez la page "ici".

### Systèmes front-end (orchestrateurs) pris en charge

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

- Kubernetes 1.17 ou version ultérieure (dernière version : 1.22)
- Mirantis Kubernetes Engine 3.4
- OpenShift 4.7, 4.8, 4.9 (derniers 4.9)

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

- Kubernetes 1.17 ou version ultérieure (dernière version : 1.22)
- OpenShift 4.7, 4.8, 4.9 (derniers 4.9)

> Les utilisateurs de Red Hat OpenShift Container Platform peuvent observer que leur fichier Initiatorname.iscsi est vide si une version inférieure à 4.6.8 est utilisée. Il s'agit d'un bug identifié par RedHat devant être corrigé avec OpenShift 4.6.8. Voir ceci "annonce de correction de bogues". NetApp vous recommande d'utiliser Astra Trident sur OpenShift 4.7 et versions ultérieures.

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

## 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
- Magasin de données Astra
- Cloud Volumes ONTAP
- Cloud Volumes Service pour AWS
- Cloud Volumes Service pour GCP
- FAS/AFF/Select 9.3 ou version ultérieure
- Baie SAN 100 % Flash (ASA) de NetApp
- Logiciel NetApp HCI/Element 8 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 | 1.17 et versions ultérieures | Non |
| Snapshots de volume | 1.17 et versions ultérieures | Non |
| Volume persistant à partir des copies Snapshot des volumes | 1.17 et versions ultérieures | Non |
| Redimensionnement PV iSCSI | 1.17 et versions ultérieures | Non |
| Chap bidirectionnel ONTAP | 1.17 et versions ultérieures | Non |
| Règles d'exportation dynamiques | 1.17 et versions ultérieures | Non |
| Opérateur Trident | 1.17 et versions ultérieures | Non |
| Préparation automatique du nœud de travail (bêta) | 1.17 et versions ultérieures | Non |
| Topologie CSI | 1.17 et versions ultérieures | Non |

## Systèmes d'exploitation hôtes testés

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.

Bien que l'Astra Trident ne « prend pas officiellement en charge » les systèmes d'exploitation spécifiques, les

distributions Linux suivantes sont connues pour fonctionner :

- Red Hat CoreOS 4.2 et 4.3
- RHEL ou CentOS 7.4 ou version ultérieure
- Ubuntu 18.04 ou version ultérieure

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

## Configuration de l'hôte

En fonction du ou des back-end utilisés, des utilitaires NFS et/ou iSCSI doivent être installés sur tous les employés du cluster. Voir "ici" pour en savoir plus.

## 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. Voir "ici" pour plus d'informations.

## 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 |
|---|---|
| v1.17.0 | • netapp/trident :21.10.0<br>• netapp/trident-autosupport :21.10<br>• k8s.gcr.io/sig-storage/csi-provisionneur:v2.2<br>• k8s.gcr.io/sig-storage/csi-attaché:v3.3.0<br>• k8s.gcr.io/sig-storage/csi-resizer:v1.3.0<br>• k8s.gcr.io/sig-storage/csi-snapshotter:v3.0.3<br>• k8s.gcr.io/sig-storage/csi-node-driver-registry:v2.3.0<br>• opérateur netapp/trident :21.10.0 (en option) |
| v1.18.0 | • netapp/trident :21.10.0<br>• netapp/trident-autosupport :21.10<br>• k8s.gcr.io/sig-storage/csi-provisionneur:v2.2<br>• k8s.gcr.io/sig-storage/csi-attaché:v3.3.0<br>• k8s.gcr.io/sig-storage/csi-resizer:v1.3.0<br>• k8s.gcr.io/sig-storage/csi-snapshotter:v3.0.3<br>• k8s.gcr.io/sig-storage/csi-node-driver-registry:v2.3.0<br>• opérateur netapp/trident :21.10.0 (en option) |

| Version Kubernetes | Image de conteneur |
| --- | --- |
| v1.19.0 | • netapp/trident :21.10.0<br>• netapp/trident-autosupport :21.10<br>• k8s.gcr.io/sig-storage/csi-provisionneur:v2.2<br>• k8s.gcr.io/sig-storage/csi-attaché:v3.3.0<br>• k8s.gcr.io/sig-storage/csi-resizer:v1.3.0<br>• k8s.gcr.io/sig-storage/csi-snapshotter:v3.0.3<br>• k8s.gcr.io/sig-storage/csi-node-driver-registry:v2.3.0<br>• opérateur netapp/trident :21.10.0 (en option) |
| v1.20.0 | • netapp/trident :21.10.0<br>• netapp/trident-autosupport :21.10<br>• k8s.gcr.io/sig-storage/csi-provisionneur:v3.0.0<br>• k8s.gcr.io/sig-storage/csi-attaché:v3.3.0<br>• k8s.gcr.io/sig-storage/csi-resizer:v1.3.0<br>• k8s.gcr.io/sig-storage/csi-snapshotter:v3.0.3<br>• k8s.gcr.io/sig-storage/csi-node-driver-registry:v2.3.0<br>• opérateur netapp/trident :21.10.0 (en option) |
| v1.21.0 | • netapp/trident :21.10.0<br>• netapp/trident-autosupport :21.10<br>• k8s.gcr.io/sig-storage/csi-provisionneur:v3.0.0<br>• k8s.gcr.io/sig-storage/csi-attaché:v3.3.0<br>• k8s.gcr.io/sig-storage/csi-resizer:v1.3.0<br>• k8s.gcr.io/sig-storage/csi-snapshotter:v3.0.3<br>• k8s.gcr.io/sig-storage/csi-node-driver-registry:v2.3.0<br>• opérateur netapp/trident :21.10.0 (en option) |

| Version Kubernetes | Image de conteneur |
| --- | --- |
| v1.22.0 | • netapp/trident :21.10.0<br>• netapp/trident-autosupport :21.10<br>• k8s.gcr.io/sig-storage/csi-provisionneur:v3.0.0<br>• k8s.gcr.io/sig-storage/csi-attaché:v3.3.0<br>• k8s.gcr.io/sig-storage/csi-resizer:v1.3.0<br>• k8s.gcr.io/sig-storage/csi-snapshotter:v3.0.3<br>• k8s.gcr.io/sig-storage/csi-node-driver-registry:v2.3.0<br>• opérateur netapp/trident :21.10.0 (en option) |

Sur Kubernetes version 1.20 et ultérieure, utilisez la `k8s.gcr.io/sig-storage/csi-snapshotter:v4.x` image uniquement si `v1` la version sert le `volumesnapshots.snapshot.storage.k8s.io` CRD. Si le `v1beta1` La version sert le CRD avec/sans le `v1` utilisez la version validée `k8s.gcr.io/sig-storage/csi-snapshotter:v3.x` image.

# Présentation du déploiement

Vous pouvez déployer Astra Trident avec l'opérateur Trident ou avec `tridentctl`.

## Choisissez la méthode de déploiement

Pour déterminer la méthode de déploiement à utiliser, prenez en compte les points suivants :

### Pourquoi devrais-je utiliser l'opérateur Trident ?

Le "Opérateur Trident" Est un excellent moyen de gérer de façon dynamique les ressources d'Astra Trident et d'automatiser la phase de configuration. Certaines conditions préalables doivent être satisfaites. Voir "les conditions requises".

Le conducteur de Trident offre plusieurs avantages comme décrit ci-dessous.

#### Fonctionnalité d'auto-rétablissement

Vous pouvez surveiller une installation Trident d'Astra et prendre activement des mesures pour résoudre les problèmes, comme par exemple lorsque le déploiement est supprimé ou modifié par erreur. Lorsque l'opérateur est configuré comme déploiement, un `trident-operator-<generated-id>` le pod est créé. Ce pod associe un `TridentOrchestrator` Le système CR avec une installation Astra Trident et s'assure toujours qu'il n'y en a qu'un seul actif `TridentOrchestrator`. En d'autres termes, l'opérateur s'assure 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 idempuissante. 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.

#### Mises à jour faciles des installations existantes

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.

#### Prise en charge automatique des mises 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 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.

### Pourquoi utiliser Helm ?

Si vous utilisez Helm pour gérer d'autres applications en utilisant Helm, à partir d'Astra Trident 21.01, vous pouvez également gérer votre déploiement à l'aide de Helm.

### Quand dois-je utiliser `tridenctl`?

Si vous disposez d'un déploiement existant qui doit être mis à niveau vers ou si vous souhaitez personnaliser fortement votre déploiement, vous devez utiliser "tridentctl". Il s'agit de la méthode classique de déploiement d'Astra Trident.

## Considérations relatives au passage d'une méthode de déploiement à l'autre

Il est difficile d'imaginer un scénario dans lequel il serait souhaitable de passer d'une méthode de déploiement à l'autre. Vous devez tenir compte des éléments suivants avant d'essayer de passer d'un à un `tridentctl` déploiement vers un déploiement basé sur l'opérateur ou vice-versa :

- Utilisez toujours la même méthode pour 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 `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.

## Comprendre les modes de déploiement

Il existe trois façons de déployer Astra Trident.

### Déploiement standard

Le déploiement de Trident sur un cluster Kubernetes se traduit par deux étapes du programme d'installation d'Astra Trident :

- Récupération des images du conteneur sur Internet
- Création d'un ensemble de déploiement et/ou de diaboset de nœuds, qui fait tourner les pods Astra Trident sur tous les nœuds éligibles du cluster Kubernetes.

Pour ce faire, un déploiement standard peut être effectué de deux manières différentes :

- À l'aide de `tridentctl install`
- Utilisation de l'opérateur Trident. Vous pouvez déployer l'opérateur Trident manuellement ou à l'aide de Helm.

Ce mode d'installation est le moyen le plus simple d'installer Astra Trident et fonctionne pour la plupart des environnements qui n'imposent pas de restrictions de réseau.

### Déploiement hors ligne

Pour effectuer un déploiement pneumatique, vous pouvez utiliser le `--image-registry` indicateur lors de l'appel `tridentctl install` pour pointer vers un registre d'images privées. Si vous déployez avec l'opérateur Trident, vous pouvez également spécifier `spec.imageRegistry` dans votre `TridentOrchestrator`. Ce registre doit contenir le "Image Trident", le "Image AutoSupport Trident", Et les images CSI sidecar comme requis par votre version Kubernetes.

Pour personnaliser votre déploiement, vous pouvez utiliser `tridentctl` Générer les manifestes pour les ressources de 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.

Pour plus d'informations sur la personnalisation de votre déploiement, reportez-vous aux liens suivants :

- "Personnalisez votre déploiement basé sur l'opérateur"
*

Si vous utilisez un référentiel d'images privé, vous devez l'ajouter `/k8scsi` Pour les versions Kubernetes antérieures à 1.17 ou `/sig-storage` Pour les versions Kubernetes ultérieures à 1.17 à la fin de l'URL du registre privé. Lorsque vous utilisez un registre privé pour `tridentctl` déploiement, vous devez l'utiliser `--trident-image` et `--autosupport-image` en conjonction avec `--image-registry`. Si vous déployez Astra Trident à l'aide de l'opérateur Trident, assurez-vous que le CR orchestrator est inclus `tridentImage` et `autosupportImage` dans les paramètres d'installation.

**Déploiement à distance**

Voici une présentation générale du processus de déploiement 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.

# Déployez-le avec l'opérateur Trident

Vous pouvez déployer Astra Trident avec l'opérateur Trident. Vous pouvez déployer l'opérateur Trident manuellement ou à l'aide de Helm.

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

**Ce dont vous avez besoin**

Pour déployer Astra Trident, les prérequis suivants doivent être respectés :

- Vous disposez de privilèges complets pour un cluster Kubernetes pris en charge exécutant Kubernetes 1.17 ou une version ultérieure.
- Vous avez accès à un système de stockage NetApp pris en charge.
- Vous avez la possibilité de monter des volumes à partir de tous les nœuds workers Kubernetes.
- Vous avez un hôte Linux avec `kubectl` (ou `oc`, Si vous utilisez OpenShift) installé et configuré pour gérer le cluster Kubernetes que vous souhaitez utiliser.
- Vous avez défini le `KUBECONFIG` Variable d'environnement qui pointe vers votre configuration de cluster Kubernetes.
- Vous avez activé "Portails requis par Astra Trident".
- Si vous utilisez Kubernetes avec Docker Enterprise, "Suivez les étapes indiquées pour activer l'accès à l'interface de ligne de commande".

Vous avez tout ça ? Parfait ! Nous allons commencer.

## Déployer l'opérateur Trident à l'aide de Helm

Effectuer les étapes énumérées pour déployer l'opérateur Trident à l'aide de Helm.

**Ce dont vous avez besoin**

En plus des prérequis répertoriés ci-dessus, pour déployer l'opérateur Trident à l'aide de Helm, vous devez :

- Kubernetes 1.17 et versions ultérieures
- Version 3 de Helm

**Étapes**

1. Téléchargez le programme d'installation à partir du "GitHub pour Trident" page. L'ensemble de l'installateur comprend le graphique Helm du `/helm` répertoire.
2. Utilisez le `helm install` et spécifiez un nom pour votre déploiement. Voir l'exemple suivant :

   ```
   helm install <name> trident-operator-21.07.1.tgz --namespace <namespace
   you want to use for Trident>
   ```

   Si vous n'avez pas encore créé de namespace pour Trident, vous pouvez ajouter le `--create -namespace` paramètre au `helm install` commande. Helm crée ensuite automatiquement l'espace de noms pour vous.

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

   - `--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 :

```
$ helm install <name> trident-operator-21.07.1.tgz --set tridentDebug=true
```

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

`helm list` affiche des informations détaillées sur l'installation, telles que nom, espace de noms, graphique, état, version de l'application, numéro de révision, etc.

## Déployez l'opérateur Trident manuellement

Effectuer les étapes énumérées pour déployer manuellement l'opérateur Trident.

### Étape 1 : qualifier le cluster Kubernetes

La première chose à faire est de se connecter à l'hôte Linux et de vérifier qu'il gère un *working*, "Cluster Kubernetes pris en charge" que vous disposez des privilèges nécessaires à.

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`.

Pour vérifier si votre version de Kubernetes est ultérieure à 1.17, exécutez la commande suivante :

```
kubectl version
```

Pour vérifier si vous disposez des privilèges d'administrateur de cluster Kubernetes, exécutez la commande suivante :

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

Pour vérifier si vous pouvez lancer un pod qui utilise une image de Docker Hub et atteindre votre système de stockage sur le réseau pod, exécutez la commande suivante :

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

**Étape 2 : télécharger et configurer l'opérateur**

Depuis la version 21.01, l'opérateur de Trident se trouve dans le périmètre du cluster. Pour installer Trident, vous devez créer le `TridentOrchestrator` Définition de ressource personnalisée (CRD) et définition d'autres ressources. Vous devez effectuer ces étapes pour configurer l'opérateur avant de pouvoir installer Astra Trident.

1. Téléchargez la dernière version du "Pack d'installation Trident" À partir de la section *Downloads* et extrayez-la.

   ```
   wget https://github.com/NetApp/trident/releases/download/v21.04/trident-installer-21.04.tar.gz
   tar -xf trident-installer-21.04.tar.gz
   cd trident-installer
   ```

2. Utilisez le manifeste CRD approprié pour créer le `TridentOrchestrator` CRD. Vous créez ensuite un `TridentOrchestrator` Ressource personnalisée plus tard pour instancier une installation par l'opérateur.

   Exécutez la commande suivante :

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

3. Après le `TridentOrchestrator` Le service CRD est créé et crée les ressources suivantes nécessaires au déploiement de l'opérateur :

   - Un ServiceAccount pour l'opérateur
   - A ClusterRole et ClusterRoleBinding au ServiceAccount
   - Une stratégie de sécurité de PodSecurityPolicy dédiée
   - L'opérateur lui-même

   Le programme d'installation Trident contient des manifestes pour définir ces ressources. Par défaut, l'opérateur est déployé dans le `trident` espace de noms. Si le `trident` l'espace de noms n'existe pas, utilisez le manifeste suivant pour en créer un.

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

4. Pour déployer l'opérateur dans un espace de noms autre que celui par défaut `trident` espace de noms, vous devez mettre à jour le `serviceaccount.yaml`, `clusterrolebinding.yaml` et `operator.yaml` manifeste et génère votre `bundle.yaml`.

   Exécutez la commande suivante pour mettre à jour les manifestes YAML et générer votre `bundle.yaml` à l'aide du `kustomization.yaml`:

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

   Exécutez la commande suivante pour créer les ressources et déployer l'opérateur :

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

5. Pour vérifier l'état de l'opérateur après le déploiement, procédez comme suit :

```
$ kubectl get deployment -n <operator-namespace>
NAME               READY   UP-TO-DATE   AVAILABLE   AGE
trident-operator   1/1     1            1           3m

$ kubectl get pods -n <operator-namespace>
NAME                                READY   STATUS              RESTARTS
AGE
trident-operator-54cb664d-lnjxh   1/1     Running             0
3m
```

Le déploiement de l'opérateur a réussi à créer un pod exécuté sur l'un des nœuds worker de votre cluster.

> 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 3 : créer `TridentOrchestrator` Et installer Trident**

Vous êtes maintenant prêt à installer Astra Trident avec l'opérateur ! Cela nécessitera la création `TridentOrchestrator`. Le programme d'installation Trident est fourni avec des exemples de définitions à créer `TridentOrchestrator`. Cela déclenche une installation dans le `trident` espace de noms.

```
$ 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:21.04
    Autosupport Proxy:
    Autosupport Serial Number:
    Debug:                     true
    Enable Node Prep:          false
    Image Pull Secrets:
    Image Registry:
    k8sTimeout:           30
    Kubelet Dir:          /var/lib/kubelet
    Log Format:           text
    Silence Autosupport:  false
    Trident Image:        netapp/trident:21.04.0
  Message:                  Trident installed  Namespace:
trident
  Status:                   Installed
  Version:                  v21.04.0
Events:
    Type Reason Age From Message ---- ------ ---- ---- -------Normal
    Installing 74s trident-operator.netapp.io Installing Trident Normal
    Installed 67s trident-operator.netapp.io Trident installed
```

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

`TridentOrchestrator` spécifications Voir "Personnalisez votre déploiement Trident".

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

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

Pendant l'installation, l'état de `TridentOrchestrator` modifications de `Installing` à `Installed`. Si vous observez l' `Failed` statut et l'opérateur ne peut pas récupérer lui-même, il est recommandé de vérifier les journaux de l'opérateur. Voir la "dépannage" section.

Vous pouvez vérifier que l'installation d'Astra Trident est terminée en consultant les pods qui ont été créés :

```
$ kubectl get pod -n trident
NAME                                READY   STATUS    RESTARTS   AGE
trident-csi-7d466bf5c7-v4cpw        5/5     Running   0          1m
trident-csi-mr6zc                   2/2     Running   0          1m
trident-csi-xrp7w                   2/2     Running   0          1m
trident-csi-zh2jt                   2/2     Running   0          1m
trident-operator-766f7b8658-ldzsv   1/1     Running   0          3m
```

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

```
$ ./tridentctl -n trident version
+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 21.04.0        | 21.04.0        |
+----------------+----------------+
```

Maintenant, vous pouvez avancer et créer un back-end. Voir "tâches post-déploiement".

Pour résoudre les problèmes pendant le déploiement, reportez-vous au "dépannage" section.

## Personnalisez le déploiement des opérateurs Trident

L'opérateur Trident vous permet de personnaliser l'installation d'Astra Trident à l'aide des attributs du `TridentOrchestrator` spécifications

Consultez le tableau suivant pour obtenir la liste des 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 |
| `IPv6` | Installez Astra Trident sur IPv6 | faux |
| `k8sTimeout` | Délai d'expiration pour les opérations Kubernetes | 30 secondes |
| `silenceAutosupport` | N'envoyez pas automatiquement des packs AutoSupport à NetApp | faux |
| `enableNodePrep` | Gérer automatiquement les dépendances des nœuds de travail (**BÊTA**) | faux |
| `autosupportImage` | Image conteneur pour la télémétrie AutoSupport | « netapp/trident-autosupport :21.04.0 » |
| `autosupportProxy` | Adresse/port d'un proxy pour l'envoi de télémétrie AutoSupport | "<a href="http://proxy.example.com:8888"" class="bare">http://proxy.example.com:8888"</a> |
| `uninstall` | Indicateur utilisé pour désinstaller Astra Trident | faux |
| `logFormat` | Format de connexion Astra Trident à utiliser [text,json] | « texte » |
| `tridentImage` | Image Astra Trident à installer | netapp/trident:21.04 |
| `imageRegistry` | Chemin d'accès au registre interne, du format `<registry FQDN>[:port][/subpath]` | "k8s.gcr.io/sig-storage (k8s 1.17+) ou quay.io/k8scsi" |
| `kubeletDir` | Chemin d'accès au répertoire kubelet de l'hôte | "/var/lib/kubelet" |
| `wipeout` | Liste des ressources à supprimer pour effectuer la suppression complète d'Astra Trident | |

| Paramètre | Description | Valeur par défaut |
| --- | --- | --- |
| `imagePullSecrets` | Secrets pour extraire des images d'un registre interne | |

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

La préparation automatique des nœuds worker est une fonction **bêta** destinée à être utilisée uniquement dans les environnements non-production.

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

```
$ cat deploy/crds/tridentorchestrator_cr_imagepullsecrets.yaml
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  tridentImage: netapp/trident:21.04.0
  imagePullSecrets:
  - thisisasecret
```

Si vous cherchez à personnaliser l'installation au-delà de quoi `TridentOrchestrator` les arguments permettent, vous devez envisager d'utiliser `tridentctl` Pour générer des manifestes YAML personnalisés que vous pouvez modifier si nécessaire.

# Déploiement avec tridentctl

Vous pouvez déployer Astra Trident avec un outil `tridentctl`.

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

Pour personnaliser votre déploiement, voir "ici".

**Ce dont vous avez besoin**

Pour déployer Astra Trident, les prérequis suivants doivent être respectés :

- Vous disposez de privilèges complets pour un cluster Kubernetes pris en charge.

- Vous avez accès à un système de stockage NetApp pris en charge.
- Vous avez la possibilité de monter des volumes à partir de tous les nœuds workers Kubernetes.
- Vous avez un hôte Linux avec `kubectl` (ou `oc`, Si vous utilisez OpenShift) installé et configuré pour gérer le cluster Kubernetes que vous souhaitez utiliser.
- Vous avez défini le `KUBECONFIG` Variable d'environnement qui pointe vers votre configuration de cluster Kubernetes.
- Vous avez activé "Portails requis par Astra Trident".
- Si vous utilisez Kubernetes avec Docker Enterprise, "Suivez les étapes indiquées pour activer l'accès à l'interface de ligne de commande".

Vous avez tout ça ? Parfait ! Nous allons commencer.

Pour plus d'informations sur la personnalisation de votre déploiement, reportez-vous à la section "ici".

## Étape 1 : qualifier le cluster Kubernetes

La première chose à faire est de se connecter à l'hôte Linux et de vérifier qu'il gère un *working*, "Cluster Kubernetes pris en charge" que vous disposez des privilèges nécessaires à.

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

Pour vérifier votre version de Kubernetes, exécutez la commande suivante :

```
kubectl version
```

Pour vérifier si vous disposez des privilèges d'administrateur de cluster Kubernetes, exécutez la commande suivante :

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

Pour vérifier si vous pouvez lancer un pod qui utilise une image de Docker Hub et atteindre votre système de stockage sur le réseau pod, exécutez la commande suivante :

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

Identifiez la version de votre serveur Kubernetes. Vous l'utiliserez lors de l'installation d'Astra Trident.

## Étape 2 : téléchargez et extrayez le programme d'installation

Le programme d'installation de Trident crée un pod Trident, configure les objets CRD utilisés pour maintenir son état et initialise les sidecars CSI qui effectuent des actions, tels que le provisionnement et la connexion de volumes aux hôtes du cluster.

Vous pouvez télécharger la dernière version du "Pack d'installation Trident" À partir de la section *Downloads* et extrayez-la.

Par exemple, si la dernière version est 21.07.1 :

```
wget https://github.com/NetApp/trident/releases/download/v21.07.1/trident-
installer-21.07.1.tar.gz
tar -xf trident-installer-21.07.1.tar.gz
cd trident-installer
```

## Étape 3 : installer Astra Trident

Installez Astra Trident dans l'espace de noms souhaité en exécutant le `tridentctl install` commande.

```
$ ./tridentctl install -n trident
....
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-csi-679648bd45-cv2mx
INFO Waiting for Trident REST interface.
INFO Trident REST interface is up.                 version=21.07.1
INFO Trident installation succeeded.
....
```

Cela ressemble à ceci quand le programme d'installation est terminé. Selon le nombre de nœuds du cluster Kubernetes, il est possible d'observer davantage de pods :

```
$ kubectl get pod -n trident
NAME                           READY   STATUS    RESTARTS   AGE
trident-csi-679648bd45-cv2mx   4/4     Running   0          5m29s
trident-csi-vgc8n              2/2     Running   0          5m29s

$ ./tridentctl -n trident version
+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 21.07.1        | 21.07.1        |
+----------------+----------------+
```

Si vous voyez des résultats similaires à l'exemple ci-dessus, vous avez terminé cette étape, mais Astra Trident n'est pas encore entièrement configuré. Passez à l'étape suivante. Voir "tâches post-déploiement".

Cependant, si le programme d'installation ne s'exécute pas correctement ou si vous ne voyez pas un **en cours d'exécution** `trident-csi-<generated id>`, la plate-forme n'a pas été installée.

Pour résoudre les problèmes pendant le déploiement, reportez-vous au "dépannage" section.

## Personnalisez le déploiement tridentctl

Le programme d'installation de Trident vous permet de personnaliser les attributs. Par exemple, si vous avez copié l'image Trident dans un référentiel privé, vous pouvez spécifier le nom de l'image 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]`.

Pour que Astra Trident configure automatiquement les nœuds workers pour vous, utilisez `--enable-node-prep`. Pour plus d'informations sur son fonctionnement, reportez-vous à la section "ici".

La préparation automatique des nœuds worker est une fonction **bêta** destinée à être utilisée uniquement dans les environnements non-production.

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`

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 que vous avez déployé Astra Trident, 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