



Installer à l'aide de l'opérateur Trident

Trident

NetApp
October 30, 2026

This PDF was generated from <https://docs.netapp.com/fr-fr/trident-2606/trident-install/kubernetes-deploy-operator.html> on October 30, 2026. Always check docs.netapp.com for the latest.

Sommaire

Installer à l'aide de l'opérateur Trident	1
Déployez manuellement l'opérateur Trident (mode standard)	1
Informations essentielles concernant Trident 26.06	1
Déployez manuellement l'opérateur Trident et installez Trident	1
Vérifiez l'installation	4
Déployez manuellement l'opérateur Trident (mode hors ligne)	6
Informations essentielles sur Trident	6
Déployez manuellement l'opérateur Trident et installez Trident	6
Vérifiez l'installation	10
Déployez l'opérateur Trident à l'aide de Helm (mode standard)	11
Informations essentielles concernant Trident 26.06	11
Déployez l'opérateur Trident et installez Trident à l'aide de Helm	12
Transmettez les données de configuration lors de l'installation	12
Options de configuration	13
Déployez l'opérateur Trident à l'aide de Helm (mode hors ligne)	19
Informations essentielles concernant Trident 26.06	19
Déployez l'opérateur Trident et installez Trident à l'aide de Helm	20
Transmettez les données de configuration lors de l'installation	21
Options de configuration	21
Personnaliser l'installation de l'opérateur Trident	27
Comprendre les pods de contrôleur et les pods de nœud	27
Options de configuration	27
Exemples de configurations	31

Installer à l'aide de l'opérateur Trident

Déployez manuellement l'opérateur Trident (mode standard)

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

Informations essentielles concernant Trident 26.06

Vous devez lire les informations essentielles suivantes concernant Trident.

▼ Informations essentielles sur Trident

- Kubernetes 1.37 est désormais pris en charge par Trident. Veuillez mettre à niveau Trident avant de mettre à niveau Kubernetes.
- Trident impose strictement l'utilisation de la configuration multipath dans les environnements SAN, avec une valeur recommandée de `find_multipaths: no` dans le fichier `multipath.conf`.

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

Déployez manuellement l'opérateur Trident et installez Trident

Vérifiez ["aperçu de l'installation"](#) pour vous assurer que vous avez respecté les prérequis d'installation et sélectionné la bonne option d'installation pour votre environnement.

Avant de commencer

Avant de commencer l'installation, connectez-vous à l'hôte Linux et vérifiez qu'il gère un système fonctionnel, ["cluster Kubernetes pris en charge"](#) et que vous disposez des privilèges nécessaires.



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

▼ Details

1. Vérifiez votre version de Kubernetes :

```
kubectl version
```

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

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

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

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

Étape 1 : Téléchargez le package d'installation de Trident

Le package d'installation de Trident contient tout le nécessaire pour déployer l'opérateur Trident et installer Trident. Téléchargez et extrayez la dernière version du programme d'installation Trident depuis ["la section Assets sur GitHub"](#).

```
wget https://github.com/NetApp/trident/releases/download/v26.06.0/trident-
installer-26.06.0.tar.gz
tar -xf trident-installer-26.06.0.tar.gz
cd trident-installer
```

Étape 2 : Créez le `TridentOrchestrator` CRD

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

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

Étape 3 : Déployer l'opérateur Trident

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

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

Avant de commencer

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

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

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

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

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

- b. Compilez le bundle en utilisant la commande suivante où `<bundle.yaml>` est `bundle_pre_1_25.yaml` ou `bundle_post_1_25.yaml` selon votre version de Kubernetes.

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

Étapes

1. Créer les ressources et déployer l'opérateur :

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

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

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



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

Étape 4 : Créez le `TridentOrchestrator` et installez Trident

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

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

kubectl describe torc trident

Name:          trident
Namespace:
Labels:        <none>
Annotations:   <none>
API Version:   trident.netapp.io/v1
Kind:          TridentOrchestrator
...
Spec:
  Debug:      true
  Namespace:  trident
```

```

nodePrep:
- iscsi
Status:
  Current Installation Params:
    IPv6: false
    Autosupport Hostname:
    Autosupport Image: netapp/trident-autosupport:26.06
    Autosupport Proxy:
    Autosupport Serial Number:
    Debug: true
    Image Pull Secrets:
    Image Registry:
    k8sTimeout: 30
    Kubelet Dir: /var/lib/kubelet
    Log Format: text
    Silence Autosupport: false
    Trident Image: netapp/trident:26.06.0
  Message: Trident installed Namespace:
trident
  Status: Installed
  Version: v26.06.0
Events:
  Type Reason Age From Message ----
  Installing 74s trident-operator.netapp.io Installing Trident Normal
  Installed 67s trident-operator.netapp.io Trident installed

```

Vérifiez l'installation

Il existe plusieurs façons de vérifier votre installation.

Utilisation `TridentOrchestrator` du statut

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

Statut	Description
Installation	L'opérateur installe Trident à l'aide de ce <code>TridentOrchestrator</code> CR.
Installé	Trident a été installé avec succès.
Désinstallation	L'opérateur désinstalle Trident, parce que <code>spec.uninstall=true</code> .
Désinstallé	Trident est désinstallé.

Statut	Description
Échec	L'opérateur n'a pas pu installer, corriger, mettre à jour ou désinstaller Trident ; l'opérateur essaiera automatiquement de se rétablir de cet état. Si cet état persiste, un dépannage sera nécessaire.
Mise à jour	L'opérateur met à jour une installation existante.
Erreur	Le TridentOrchestrator n'est pas utilisé. Un autre existe déjà.

Utilisation de l'état de création du pod

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

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

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

En utilisant tridentctl

Vous pouvez utiliser `tridentctl` pour vérifier la version de Trident installée.

```
./tridentctl -n trident version
```

```
+-----+
| SERVER VERSION | CLIENT VERSION |
+-----+
| 26.06.0       | 26.06.0       |
+-----+
```

Déployez manuellement l'opérateur Trident (mode hors ligne)

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

Informations essentielles sur Trident

Vous devez lire les informations essentielles suivantes concernant Trident.

▼ Informations essentielles sur Trident

- Kubernetes 1.37 est désormais pris en charge par Trident. Veuillez mettre à niveau Trident avant de mettre à niveau Kubernetes.
- Trident impose strictement l'utilisation de la configuration multipath dans les environnements SAN, avec une valeur recommandée de `find_multipaths: no` dans le fichier `multipath.conf`.

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

Déployez manuellement l'opérateur Trident et installez Trident

Vérifiez ["aperçu de l'installation"](#) pour vous assurer que vous avez respecté les prérequis d'installation et sélectionné la bonne option d'installation pour votre environnement.

Avant de commencer

Connectez-vous à l'hôte Linux et vérifiez qu'il gère un système fonctionnel et ["cluster Kubernetes pris en charge"](#) que vous disposez des privilèges nécessaires.



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

▼ Details

1. Vérifiez votre version de Kubernetes :

```
kubectl version
```

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

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

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

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

Étape 1 : Téléchargez le package d'installation de Trident

Le package d'installation de Trident contient tout le nécessaire pour déployer l'opérateur Trident et installer Trident. Téléchargez et extrayez la dernière version du programme d'installation Trident depuis ["la section Assets sur GitHub"](#).

```
wget https://github.com/NetApp/trident/releases/download/v6.0/trident-
installer-26.06.0.tar.gz
tar -xf trident-installer-26.06.0.tar.gz
cd trident-installer
```

Étape 2 : Créez le `TridentOrchestrator` CRD

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

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

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

Dans `/deploy/operator.yaml`, mettez à jour `image: docker.io/netapp/trident-operator:26.06.0` pour refléter l'emplacement de votre registre d'images. Votre ["Images de Trident et de CSI"](#) peut être situé dans un registre ou dans des registres différents, mais toutes les images CSI doivent être situées dans le même registre. Par exemple :

- `image: <your-registry>/trident-operator:26.06.0` si toutes vos images sont situées dans un seul registre.
- `image: <your-registry>/netapp/trident-operator:26.06.0` si votre image Trident se trouve dans un registre différent de celui de vos images CSI.

Étape 4 : Déployer l'opérateur Trident

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

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

Avant de commencer

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

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

- Pour déployer l'opérateur dans un espace de noms autre que l'`trident`namespace, mettez à jour `serviceaccount.yaml`, `clusterrolebinding.yaml` et `operator.yaml` et générez votre fichier bundle en utilisant le `kustomization.yaml`.
 - a. Créez le `kustomization.yaml` en utilisant la commande suivante où `<bundle.yaml>` est `bundle_pre_1_25.yaml` ou `bundle_post_1_25.yaml` selon votre version de Kubernetes.

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

- b. Compilez le bundle en utilisant la commande suivante où `<bundle.yaml>` est `bundle_pre_1_25.yaml` ou `bundle_post_1_25.yaml` selon votre version de Kubernetes.

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

Étapes

1. Créer les ressources et déployer l'opérateur :

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

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

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



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

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

Votre "[Images de Trident et de CSI](#)" peut être situé dans un registre ou dans des registres différents, mais toutes les images CSI doivent être situées dans le même registre. Mettez à jour `deploy/crds/tridentorchestrator_cr.yaml` pour ajouter les spécifications d'emplacement supplémentaires en fonction de la configuration de votre registre.

Images dans un seul registre

```
imageRegistry: "<your-registry>"
autosupportImage: "<your-registry>/trident-autosupport:26.06"
```

```
tridentImage: "<your-registry>/trident:26.06.0"
```

Images dans différents registres

```
imageRegistry: "<your-registry>"
autosupportImage: "<your-registry>/trident-autosupport:26.06"
tridentImage: "<your-registry>/trident:26.06.0"
```

Étape 6 : Créez le TridentOrchestrator et installez Trident

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

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

kubectl describe torc trident

Name:          trident
Namespace:
Labels:        <none>
Annotations:   <none>
API Version:   trident.netapp.io/v1
Kind:          TridentOrchestrator
...
Spec:
  Autosupport Image:  <your-registry>/trident-autosupport:26.06
  Debug:              true
  Image Registry:     <your-registry>
  Namespace:          trident
  Trident Image:      <your-registry>/trident:26.06.0
Status:
  Current Installation Params:
    IPv6:              false
    Autosupport Hostname:
    Autosupport Image:  <your-registry>/trident-autosupport:26.06
    Autosupport Proxy:
    Autosupport Serial Number:
    Debug:              true
    Http Request Timeout: 90s
    Image Pull Secrets:
    Image Registry:     <your-registry>
    k8sTimeout:         30
```

```

Kubelet Dir:          /var/lib/kubelet
Log Format:           text
Probe Port:          17546
Silence Autosupport: false
Trident Image:       <your-registry>/trident:26.06.0
Message:             Trident installed
Namespace:           trident
Status:              Installed
Version:             v26.06.0
Events:
  Type Reason Age From Message ----
  Installing 74s trident-operator.netapp.io Installing Trident Normal
  Installed 67s trident-operator.netapp.io Trident installed

```

Vérifiez l'installation

Il existe plusieurs façons de vérifier votre installation.

Utilisation `TridentOrchestrator` du statut

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

Statut	Description
Installation	L'opérateur installe Trident à l'aide de ce <code>TridentOrchestrator</code> CR.
Installé	Trident a été installé avec succès.
Désinstallation	L'opérateur désinstalle Trident, parce que <code>spec.uninstall=true</code> .
Désinstallé	Trident est désinstallé.
Échec	L'opérateur n'a pas pu installer, corriger, mettre à jour ou désinstaller Trident ; l'opérateur essaiera automatiquement de se rétablir de cet état. Si cet état persiste, un dépannage sera nécessaire.
Mise à jour	L'opérateur met à jour une installation existante.
Erreur	Le <code>TridentOrchestrator</code> n'est pas utilisé. Un autre existe déjà.

Utilisation de l'état de création du pod

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

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

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

En utilisant `tridentctl`

Vous pouvez utiliser `tridentctl` pour vérifier la version de Trident installée.

```
./tridentctl -n trident version
```

```
+-----+-----+
| SERVER VERSION | CLIENT VERSION |
+-----+-----+
| 26.06.0       | 26.06.0       |
+-----+-----+
```

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

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

Informations essentielles concernant Trident 26.06

Vous devez lire les informations essentielles suivantes concernant Trident.

▼ **Informations essentielles sur Trident**

- Kubernetes 1.37 est désormais pris en charge par Trident. Veuillez mettre à niveau Trident avant de mettre à niveau Kubernetes.
- Trident impose strictement l'utilisation de la configuration multipath dans les environnements SAN, avec une valeur recommandée de `find_multipaths: no` dans le fichier `multipath.conf`.

L'utilisation d'une configuration sans multipath ou l'utilisation de `find_multipaths: yes` ou

`find_multipaths: smart` dans le fichier `multipath.conf` entraînera des échecs de montage. Trident a recommandé l'utilisation de `find_multipaths: no` depuis la version 21.07.

Déployez l'opérateur Trident et installez Trident à l'aide de Helm

Avec Trident "[Graphique Helm](#)" vous pouvez déployer l'opérateur Trident et installer Trident en une seule étape.

Vérifiez "[aperçu de l'installation](#)" pour vous assurer que vous avez respecté les prérequis d'installation et sélectionné la bonne option d'installation pour votre environnement.

Avant de commencer

En plus de "[prérequis de déploiement](#)" vous avez besoin de "[Helm version 3](#)".

Étapes

1. Ajoutez le dépôt Trident Helm :

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

2. Utilisez `helm install` et spécifiez un nom pour votre déploiement comme dans l'exemple suivant où `100.2606.0` est la version de Trident que vous installez.

```
helm install <name> netapp-trident/trident-operator --version 100.2606.0  
--create-namespace --namespace <trident-namespace>
```



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

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

Transmettez les données de configuration lors de l'installation

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

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

Par exemple, pour modifier la valeur par défaut de `debug`, exécutez la commande suivante où `100.2606.0` est la version de Trident que vous installez :

```
helm install <name> netapp-trident/trident-operator --version 100.2606.0
```

```
--create-namespace --namespace trident --set tridentDebug=true
```

Options de configuration

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

Option	Description	Défaut
<code>nodeSelector</code>	Étiquettes de nœud pour l'affectation des pods	
<code>podAnnotations</code>	Annotations des pods	
<code>deploymentAnnotations</code>	Annotations de déploiement	
<code>tolerations</code>	Tolérances pour l'affectation des pods	
<code>affinity</code>	Affinité pour l'affectation des pods	<pre>affinity: nodeAffinity: requiredDuringSchedulingIgnoredDuringExecution: nodeSelectorTerms: - matchExpressions: - key: kubernetes.io/arch operator: In values: - arm64 - amd64 - key: kubernetes.io/os operator: In values: - linux</pre> <div> Ne supprimez pas l'affinité par défaut du fichier <code>values.yaml</code>. Lorsque vous souhaitez fournir une affinité personnalisée, étendez l'affinité par défaut.</div>

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

Option	Description	Défaut
operatorDebug	Permet de définir le niveau de journalisation de l'opérateur Trident sur débogage.	true
operatorImage	Permet de remplacer complètement l'image pour trident-operator.	""
operatorImageTag	Permet de remplacer l'étiquette de l'`trident-operator`image.	""
tridentIPv6	Permet d'activer Trident pour fonctionner dans des clusters IPv6.	false
tridentK8sTimeout	Remplace le délai d'expiration par défaut de 30 secondes pour la plupart des opérations de l'API Kubernetes (si non nul, en secondes).	0
tridentHttpRequestTimeout	Permet de remplacer le délai d'expiration par défaut de 90 secondes pour les requêtes HTTP, 0s étant une durée infinie pour le délai d'expiration. Les valeurs négatives ne sont pas autorisées.	"90s"
tridentSilenceAutoSupport	Permet de désactiver les rapports périodiques AutoSupport de Trident.	false
tridentAutoSupportImageTag	Permet de remplacer l'étiquette de l'image pour le conteneur Trident AutoSupport.	<version>
tridentAutoSupportProxy	Permet au conteneur Trident AutoSupport de communiquer avec son serveur via un proxy HTTP.	""
tridentLogFormat	Définit le format de journalisation Trident (text ou json).	"text"
tridentDisableAuditLog	Désactive le journal d'audit Trident.	true
tridentLogLevel	Permet de définir le niveau de journalisation de Trident sur : trace, debug, info, warn, error ou fatal.	"info"

Option	Description	Défaut
tridentDebug	Permet de définir le niveau de journalisation de Trident sur debug. Vous pouvez automatiser le processus de déconnexion forcée grâce à l'intégration avec l'opérateur de vérification de l'état du nœud (NHC). Pour plus d'informations, consultez "Automatisation du basculement des applications avec état avec Trident" .	false
tridentLogWorkflows	Permet d'activer des flux de travail Trident spécifiques pour la journalisation des traces ou la suppression des journaux.	""
tridentLogLayers	Permet d'activer des couches Trident spécifiques pour la journalisation des traces ou la suppression des journaux.	""
tridentImage	Permet de remplacer complètement l'image pour Trident.	""
tridentImageTag	Permet de remplacer l'étiquette de l'image pour Trident.	""
tridentProbePort	Permet de remplacer le port par défaut utilisé pour les sondes de liveness/readiness de Kubernetes.	""
windows	Permet d'installer Trident sur un nœud de travail Windows.	false
enableForceDetach	Permet d'activer la fonction de détachement forcé.	false
excludePodSecurityPolicy	Exclut la création de la politique de sécurité du pod opérateur.	false
cloudProvider	Définissez sur "Azure" lors de l'utilisation d'identités gérées ou d'une identité cloud sur un cluster AKS. Définissez sur "AWS" lors de l'utilisation d'une identité cloud sur un cluster EKS.	""

Option	Description	Défaut
cloudIdentity	Définissez sur workload identity (« azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxx ») lors de l'utilisation de l'identité cloud sur un cluster AKS. Définissez sur AWS IAM role (« 'eks.amazonaws.com/role-arn: arn:aws:iam::123456:role/trident-role' ») lors de l'utilisation de l'identité cloud sur un cluster EKS.	""
iscsiSelfHealingInterval	Intervalle auquel l'auto-réparation iSCSI est déclenchée.	5m0s
iscsiSelfHealingWaitTime	La durée après laquelle l'auto-réparation iSCSI initie une tentative de résolution d'une session obsolète en effectuant une déconnexion puis une reconnexion.	7m0s
nodePrep	Permet à Trident de préparer les nœuds du cluster Kubernetes à gérer les volumes en utilisant le protocole de stockage des données spécifié. Actuellement, iscsi est la seule valeur prise en charge. REMARQUE : À partir de OpenShift 4.19, la version minimale de Trident prise en charge pour cette fonctionnalité est 25.06.1.	
enableConcurrency	Permet le fonctionnement simultané des contrôleurs Trident pour un débit amélioré. Pour plus d'informations sur les pilotes pris en charge, consultez le lien : Controller Scalability documentation .	false
k8sAPIQPS	Limite de requêtes par seconde (QPS) utilisée par le contrôleur lors de la communication avec le serveur d'API Kubernetes. La valeur de Burst est définie automatiquement en fonction de la valeur QPS.	100; facultatif

Option	Description	Défaut
resources	<p>Définit les limites et les demandes de ressources Kubernetes pour le contrôleur Trident, le nœud et les pods d'opérateur. Vous pouvez configurer le processeur et la mémoire de chaque conteneur et sidecar afin de gérer l'allocation des ressources dans Kubernetes.</p> <p>Pour plus d'informations sur la configuration des demandes et des limites de ressources, consultez "Gestion des ressources pour les pods et les conteneurs".</p> <div>  <ul style="list-style-type: none"> • NE MODIFIEZ PAS les noms de conteneurs ou de champs. • NE PAS modifier l'indentation - L'indentation YAML est essentielle pour une analyse correcte. • Aucune limite n'est appliquée par défaut - seules les requêtes ont des valeurs par défaut et sont automatiquement appliquées si elles ne sont pas spécifiées. • Les noms des conteneurs sont listés tels qu'ils apparaissent dans les spécifications du pod. • Les sidecars sont répertoriés sous chaque conteneur principal. Consultez le </div> <div>  </div>	<pre>resources: controller: trident-main: requests: cpu: 10m memory: 80Mi limits: cpu: memory: csi-provisioner: requests: cpu: 2m memory: 20Mi limits: cpu: memory: csi-attacher: requests: cpu: 2m memory: 20Mi limits: cpu: memory: csi-resizer: requests: cpu: 3m memory: 20Mi limits: cpu: memory: csi-snapshotter: requests: cpu: 2m memory: 20Mi limits: cpu: memory: trident-autosupport: requests: cpu: 1m memory: 30Mi limits: cpu: memory: node:</pre>

Option	Description	Défaut
httpsMetrics	Activer HTTPS pour le point de terminaison des métriques Prometheus.	false
hostNetwork	Active la mise en réseau de l'hôte pour le contrôleur Trident. Ceci est utile lorsque vous souhaitez séparer le trafic frontal et dorsal dans un réseau multi-home.	false

Comprendre les pods de contrôleur et les pods de nœud

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

Kubernetes "[sélecteurs de nœuds](#)" et "[tolérances et taints](#)" sont utilisés pour contraindre un pod à s'exécuter sur un nœud spécifique ou préféré. En utilisant le `ControllerPlugin` et `NodePlugin`, vous pouvez spécifier des contraintes et des substitutions.

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

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

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

Informations essentielles concernant Trident 26.06

Vous devez lire les informations essentielles suivantes concernant Trident.

▼ **Informations essentielles sur Trident**

- Kubernetes 1.37 est désormais pris en charge par Trident. Veuillez mettre à niveau Trident avant de mettre à niveau Kubernetes.
- Trident impose strictement l'utilisation de la configuration multipath dans les environnements SAN, avec une valeur recommandée de `find_multipaths: no` dans le fichier `multipath.conf`.

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

Déployez l'opérateur Trident et installez Trident à l'aide de Helm

Avec Trident ["Graphique Helm"](#) vous pouvez déployer l'opérateur Trident et installer Trident en une seule étape.

Vérifiez ["aperçu de l'installation"](#) pour vous assurer que vous avez respecté les prérequis d'installation et sélectionné la bonne option d'installation pour votre environnement.

Avant de commencer

En plus de ["prérequis de déploiement"](#) vous avez besoin de ["Helm version 3"](#).



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

Étapes

1. Ajoutez le dépôt Trident Helm :

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

2. Utilisez `helm install` et spécifiez un nom pour votre déploiement et l'emplacement du registre d'images. Votre ["Images de Trident et de CSI"](#) peut être situé dans un registre ou dans des registres différents, mais toutes les images CSI doivent être situées dans le même registre. Dans les exemples, 100.2606.0 correspond à la version de Trident que vous installez.

Images dans un seul registre

```
helm install <name> netapp-trident/trident-operator --version  
100.2606.0 --set imageRegistry=<your-registry> --create-namespace  
--namespace <trident-namespace> --set nodePrep={iscsi}
```

Images dans différents registres

```
helm install <name> netapp-trident/trident-operator --version  
100.2606.0 --set imageRegistry=<your-registry> --set operatorImage  
=<your-registry>/trident-operator:26.06.0 --set  
tridentAutosupportImage=<your-registry>/trident-autosupport:26.06  
--set tridentImage=<your-registry>/trident:26.06.0 --create  
-namespace --namespace <trident-namespace> --set nodePrep={iscsi}
```



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

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

Transmettez les données de configuration lors de l'installation

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

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

Par exemple, pour modifier la valeur par défaut de `debug`, exécutez la commande suivante où `100.2606.0` est la version de Trident que vous installez :

```
helm install <name> netapp-trident/trident-operator --version 100.2606.0
--create-namespace --namespace trident --set tridentDebug=true
```

Pour ajouter la valeur `nodePrep`, exécutez la commande suivante :

```
helm install <name> netapp-trident/trident-operator --version 100.2606.0
--create-namespace --namespace trident --set nodePrep={iscsi}
```

Options de configuration

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



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

Option	Description	Défaut
<code>nodeSelector</code>	Étiquettes de nœud pour l'affectation des pods	
<code>podAnnotations</code>	Annotations des pods	
<code>deploymentAnnotations</code>	Annotations de déploiement	
<code>tolerations</code>	Tolérances pour l'affectation des pods	

Option	Description	Défaut
affinity	Affinité pour l'affectation des pods	<pre> affinity: nodeAffinity: requiredDuringSchedulingIgnoredDuringExecution: nodeSelectorTerms: - matchExpressions: - key: kubernetes.io/arch operator: In values: - arm64 - amd64 - key: kubernetes.io/os operator: In values: - linux </pre> <div>  <p>Ne supprimez pas l'affinité par défaut du fichier values.yaml. Lorsque vous souhaitez fournir une affinité personnalisée, étendez l'affinité par défaut.</p> </div>
tridentControllerPluginNodeSelector	Sélecteurs de nœuds supplémentaires pour les pods. Consultez " Comprendre les pods de contrôleur et les pods de nœud " pour plus de détails.	
tridentControllerPluginTolerations	Remplace les tolérances Kubernetes pour les pods. Consultez " Comprendre les pods de contrôleur et les pods de nœud " pour plus de détails.	

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

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

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

Option	Description	Défaut
resources	<p>Définit les limites et les demandes de ressources Kubernetes pour le contrôleur Trident, le nœud et les pods d'opérateur. Vous pouvez configurer le processeur et la mémoire de chaque conteneur et sidecar afin de gérer l'allocation des ressources dans Kubernetes.</p> <p>Pour plus d'informations sur la configuration des demandes et des limites de ressources, consultez "Gestion des ressources pour les pods et les conteneurs".</p> <div>  <ul style="list-style-type: none"> • NE MODIFIEZ PAS les noms de conteneurs ou de champs. • NE PAS modifier l'indentation - L'indentation YAML est essentielle pour une analyse correcte. </div> <div>  <ul style="list-style-type: none"> • Aucune limite n'est appliquée par défaut - seules les requêtes ont des valeurs par défaut. • Les noms des conteneurs sont listés tels qu'ils apparaissent dans les spécifications du pod. • Les sidecars sont répertoriés sous chaque conteneur principal. • Consultez le champ <code>status.CurrentInstallationParams</code> </div>	<pre>resources: controller: trident-main: requests: cpu: 10m memory: 80Mi limits: cpu: memory: csi-provisioner: requests: cpu: 2m memory: 20Mi limits: cpu: memory: csi-attacher: requests: cpu: 2m memory: 20Mi limits: cpu: memory: csi-resizer: requests: cpu: 3m memory: 20Mi limits: cpu: memory: csi-snapshotter: requests: cpu: 2m memory: 20Mi limits: cpu: memory: trident-autosupport: requests: cpu: 1m memory: 30Mi limits: cpu: memory:</pre>

Personnaliser l'installation de l'opérateur Trident

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

Comprendre les pods de contrôleur et les pods de nœud

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

Kubernetes "[sélecteurs de nœuds](#)" et "[tolérances et taints](#)" sont utilisés pour contraindre un pod à s'exécuter sur un nœud spécifique ou préféré. En utilisant le `ControllerPlugin` et `NodePlugin`, vous pouvez spécifier des contraintes et des substitutions.

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

Options de configuration



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

Ce tableau détaille `TridentOrchestrator` les attributs.

Paramètre	Description	Défaut
<code>namespace</code>	Espace de noms dans lequel installer Trident	"default"
<code>debug</code>	Activer le débogage pour Trident	false
<code>enableForceDetach</code>	<code>ontap-san</code> , <code>ontap-san-economy</code> , <code>ontap-nas</code> et <code>ontap-nas-economy</code> seulement. Fonctionne avec Kubernetes Non-Graceful Node Shutdown (NGNS) pour permettre aux administrateurs de cluster de migrer en toute sécurité les charges de travail avec des volumes montés vers de nouveaux nœuds si un nœud devient défaillant. Pour plus d'informations, consultez " Automatisation du basculement des applications avec état avec Trident ".	false
<code>windows</code>	Le paramètre <code>true</code> active l'installation sur les nœuds de travail Windows.	false

Paramètre	Description	Défaut
cloudProvider	Définissez sur "Azure" lors de l'utilisation d'identités gérées ou d'une identité cloud sur un cluster AKS. Définir sur "AWS" lors de l'utilisation d'une identité cloud sur un cluster EKS. Définir sur "GCP" lors de l'utilisation d'une identité cloud sur un cluster GKE.	" "
cloudIdentity	Définissez sur workload identity (« azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx ») lors de l'utilisation de l'identité cloud sur un cluster AKS. Définissez sur AWS IAM role (« 'eks.amazonaws.com/role-arn: arn:aws:iam::123456:role/trident-role' ») lors de l'utilisation de l'identité cloud sur un cluster EKS. Définissez sur cloud identity ("iam.gke.io/gcp-service-account: xxxx@mygcpproject.iam.gserviceaccount.com ") lors de l'utilisation de l'identité cloud sur un cluster GKE.	" "
IPv6	Installer Trident sur IPv6	false
k8sTimeout	Délai d'attente pour les opérations Kubernetes. <div>  <div>Le `k8sTimeout` paramètre s'applique uniquement à l'installation de Trident.</div> </div>	180sec
silenceAutosupport	N'envoyez pas automatiquement les bundles d'autosupport à NetApp	false
autosupportImage	L'image conteneur pour la télémétrie Autosupport	"netapp/trident-autosupport10"
autosupportProxy	L'adresse/port d'un proxy pour l'envoi des données de télémétrie Autosupport	"http://proxy.example.com:8888"
uninstall	Un indicateur utilisé pour désinstaller Trident	false
logFormat	Format de journalisation Trident à utiliser [text,json]	"text"
tridentImage	Image Trident à installer	"netapp/trident:26.06"
imageRegistry	Chemin d'accès au registre interne, du format <registry FQDN>[:port] [/subpath]	"registry.k8s.io"
kubeletDir	Chemin d'accès au répertoire kubelet sur l'hôte	"/var/lib/kubelet"
wipeout	Une liste des ressources à supprimer pour effectuer une suppression complète de Trident	
imagePullSecrets	Secrets pour extraire des images d'un registre interne	
imagePullPolicy	Définit la stratégie de récupération d'images pour l'opérateur Trident. Les valeurs valides sont : Always pour toujours récupérer l'image. IfNotPresent pour récupérer l'image uniquement si elle n'existe pas déjà sur le nœud. Never pour ne jamais récupérer l'image.	IfNotPresent

Paramètre	Description	Défaut
controllerPluginNodeSelector	Sélecteurs de nœuds supplémentaires pour les pods. Suit le même format que <code>pod.spec.nodeSelector</code> .	Aucune valeur par défaut ; optionnel
controllerPluginTolerations	Remplace les tolérances Kubernetes pour les pods. Suit le même format que <code>pod.spec.Tolerations</code> .	Aucune valeur par défaut ; optionnel
nodePluginNodeSelector	Sélecteurs de nœuds supplémentaires pour les pods. Suit le même format que <code>pod.spec.nodeSelector</code> .	Aucune valeur par défaut ; optionnel
nodePluginTolerations	Remplace les tolérances Kubernetes pour les pods. Suit le même format que <code>pod.spec.Tolerations</code> .	Aucune valeur par défaut ; optionnel
nodePrep	<p>Permet à Trident de préparer les nœuds du cluster Kubernetes à gérer les volumes en utilisant le protocole de stockage des données spécifié.</p> <p>Actuellement, <code>iscsi</code> est la seule valeur prise en charge.</p> <div>  <p>À partir de OpenShift 4.19, la version minimale de Trident prise en charge pour cette fonctionnalité est 25.06.1.</p> </div>	
k8sAPIQPS	Limite de requêtes par seconde (QPS) utilisée par le contrôleur lors de la communication avec le serveur d'API Kubernetes. La valeur de Burst est définie automatiquement en fonction de la valeur QPS.	100; facultatif
enableConcurrency	Permet le fonctionnement simultané des contrôleurs Trident pour un débit amélioré. Pour plus d'informations sur les pilotes pris en charge, consultez le lien : Controller Scalability documentation .	resources

Paramètre	Description	Défaut
<p>Définit les limites et les requêtes de ressources Kubernetes pour le contrôleur Trident et les pods de nœuds. Vous pouvez configurer le processeur et la mémoire de chaque conteneur et sidecar afin de gérer l'allocation des ressources dans Kubernetes.</p> <p>Pour plus d'informations sur la configuration des demandes et des limites de ressources, consultez "Gestion des ressources pour les pods et les conteneurs".</p> <div> <div>  </div> <div> <ul style="list-style-type: none"> • NE MODIFIEZ PAS les noms de conteneurs ou de champs . • NE PAS modifier l'indentation - L'indentation YAML est essentielle pour une analyse correcte. </div> </div> <div> <div>  </div> <div> <ul style="list-style-type: none"> • Aucune limite n'est </div> </div>	<pre> resources: controller: trident-main: requests: cpu: 10m memory: 80Mi limits: cpu: memory: csi-provisioner: requests: cpu: 2m memory: 20Mi limits: cpu: memory: csi-attacher: requests: cpu: 2m memory: 20Mi limits: cpu: memory: csi-resizer: requests: cpu: 3m memory: 20Mi limits: cpu: memory: csi-snapshotter: requests: cpu: 2m memory: 20Mi limits: cpu: memory: trident-autosupport: requests: cpu: 1m memory: 30Mi limits: cpu: memory: node: </pre>	httpsMetrics

Paramètre	Description	Défaut
Activer HTTPS pour le point de terminaison des métriques Prometheus.	false	hostNetwork



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

Exemples de configurations

Vous pouvez utiliser les attributs dans [Options de configuration](#) lors de la définition `TridentOrchestrator` pour personnaliser votre installation.

▼ Configuration personnalisée de base

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

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

▼ Sélecteurs de nœuds

Cet exemple installe Trident avec des sélecteurs de nœuds.

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

▼ Nœuds de travail Windows

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

deploy/crds/tridentorchestrator_cr.yaml, installe Trident sur un nœud de travail Windows.

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

▼ Identités gérées sur un cluster AKS

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

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

▼ Identité cloud sur un cluster AKS

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

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

▼ Identité cloud sur un cluster EKS

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

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
```

```

name: trident
spec:
  debug: true
  namespace: trident
  cloudProvider: "AWS"
  cloudIdentity: "'eks.amazonaws.com/role-arn:
arn:aws:iam::123456:role/trident-role'"

```

▼ Identité cloud pour GKE

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

```

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

```

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

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



Avertissement : Les valeurs de requête et de limite fournies dans cet exemple sont données à titre indicatif uniquement. Ajustez ces valeurs en fonction de votre environnement et de vos besoins en charge de travail.

```

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

```

```
    requests:
      cpu: 10m
      memory: 80Mi
    limits:
      cpu: 200m
      memory: 256Mi
# sidecars
csi-provisioner:
  requests:
    cpu: 2m
    memory: 20Mi
  limits:
    cpu: 100m
    memory: 64Mi
csi-attacher:
  requests:
    cpu: 2m
    memory: 20Mi
  limits:
    cpu: 100m
    memory: 64Mi
csi-resizer:
  requests:
    cpu: 3m
    memory: 20Mi
  limits:
    cpu: 100m
    memory: 64Mi
csi-snapshotter:
  requests:
    cpu: 2m
    memory: 20Mi
  limits:
    cpu: 100m
    memory: 64Mi
trident-autosupport:
  requests:
    cpu: 1m
    memory: 30Mi
  limits:
    cpu: 50m
    memory: 128Mi
node:
  linux:
    trident-main:
      requests:
```

```

        cpu: 10m
        memory: 60Mi
    limits:
        cpu: 200m
        memory: 256Mi
# sidecars
node-driver-registrar:
    requests:
        cpu: 1m
        memory: 10Mi
    limits:
        cpu: 50m
        memory: 32Mi

```

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

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



Avertissement : Les valeurs de requête et de limite fournies dans cet exemple sont données à titre indicatif uniquement. Ajustez ces valeurs en fonction de votre environnement et de vos besoins en charge de travail.

```

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

```

```

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

```

```
    limits:
      cpu: 50m
      memory: 32Mi
windows:
  trident-main:
    requests:
      cpu: 6m
      memory: 40Mi
    limits:
      cpu: 200m
      memory: 128Mi
# sidecars
node-driver-registrar:
  requests:
    cpu: 6m
    memory: 40Mi
  limits:
    cpu: 100m
    memory: 128Mi
liveness-probe:
  requests:
    cpu: 2m
    memory: 40Mi
  limits:
    cpu: 50m
    memory: 64Mi
```

Informations sur le copyright

Copyright © 2026 NetApp, Inc. Tous droits réservés. Imprimé aux États-Unis. Aucune partie de ce document protégé par copyright ne peut être reproduite sous quelque forme que ce soit ou selon quelque méthode que ce soit (graphique, électronique ou mécanique, notamment par photocopie, enregistrement ou stockage dans un système de récupération électronique) sans l'autorisation écrite préalable du détenteur du droit de copyright.

Les logiciels dérivés des éléments NetApp protégés par copyright sont soumis à la licence et à l'avis de non-responsabilité suivants :

CE LOGICIEL EST FOURNI PAR NETAPP « EN L'ÉTAT » ET SANS GARANTIES EXPRESSES OU TACITES, Y COMPRIS LES GARANTIES TACITES DE QUALITÉ MARCHANDE ET D'ADÉQUATION À UN USAGE PARTICULIER, QUI SONT EXCLUES PAR LES PRÉSENTES. EN AUCUN CAS NETAPP NE SERA TENU POUR RESPONSABLE DE DOMMAGES DIRECTS, INDIRECTS, ACCESSOIRES, PARTICULIERS OU EXEMPLAIRES (Y COMPRIS L'ACHAT DE BIENS ET DE SERVICES DE SUBSTITUTION, LA PERTE DE JOUISSANCE, DE DONNÉES OU DE PROFITS, OU L'INTERRUPTION D'ACTIVITÉ), QUELLES QU'EN SOIENT LA CAUSE ET LA DOCTRINE DE RESPONSABILITÉ, QU'IL S'AGISSE DE RESPONSABILITÉ CONTRACTUELLE, STRICTE OU DÉLICTEUELLE (Y COMPRIS LA NÉGLIGENCE OU AUTRE) DÉCOULANT DE L'UTILISATION DE CE LOGICIEL, MÊME SI LA SOCIÉTÉ A ÉTÉ INFORMÉE DE LA POSSIBILITÉ DE TELS DOMMAGES.

NetApp se réserve le droit de modifier les produits décrits dans le présent document à tout moment et sans préavis. NetApp décline toute responsabilité découlant de l'utilisation des produits décrits dans le présent document, sauf accord explicite écrit de NetApp. L'utilisation ou l'achat de ce produit ne concède pas de licence dans le cadre de droits de brevet, de droits de marque commerciale ou de tout autre droit de propriété intellectuelle de NetApp.

Le produit décrit dans ce manuel peut être protégé par un ou plusieurs brevets américains, étrangers ou par une demande en attente.

LÉGENDE DE RESTRICTION DES DROITS : L'utilisation, la duplication ou la divulgation par le gouvernement sont sujettes aux restrictions énoncées dans le sous-paragraphe (b)(3) de la clause Rights in Technical Data-Noncommercial Items du DFARS 252.227-7013 (février 2014) et du FAR 52.227-19 (décembre 2007).

Les données contenues dans les présentes se rapportent à un produit et/ou service commercial (tel que défini par la clause FAR 2.101). Il s'agit de données propriétaires de NetApp, Inc. Toutes les données techniques et tous les logiciels fournis par NetApp en vertu du présent Accord sont à caractère commercial et ont été exclusivement développés à l'aide de fonds privés. Le gouvernement des États-Unis dispose d'une licence limitée irrévocable, non exclusive, non cessible, non transférable et mondiale. Cette licence lui permet d'utiliser uniquement les données relatives au contrat du gouvernement des États-Unis d'après lequel les données lui ont été fournies ou celles qui sont nécessaires à son exécution. Sauf dispositions contraires énoncées dans les présentes, l'utilisation, la divulgation, la reproduction, la modification, l'exécution, l'affichage des données sont interdits sans avoir obtenu le consentement écrit préalable de NetApp, Inc. Les droits de licences du Département de la Défense du gouvernement des États-Unis se limitent aux droits identifiés par la clause 252.227-7015(b) du DFARS (février 2014).

Informations sur les marques commerciales

NETAPP, le logo NETAPP et les marques citées sur le site <http://www.netapp.com/TM> sont des marques déposées ou des marques commerciales de NetApp, Inc. Les autres noms de marques et de produits sont des marques commerciales de leurs propriétaires respectifs.