Installer à l'aide de l'opérateur Trident

Trident

NetApp
July 01, 2026

Sommaire

Installer à l'aide de l'opérateur Trident

Déployez manuellement l'opérateur Trident (mode standard)

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

Informations essentielles concernant Trident 26.02

Vous devez lire les informations essentielles suivantes concernant Trident.

Informations essentielles sur Trident

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

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

Déployez manuellement l'opérateur Trident et installez Trident

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

Avant de commencer

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

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

1. Vérifiez votre version de Kubernetes :

```
kubectl version
```

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

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

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

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

Étape 1 : Téléchargez le package d'installation de Trident

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

```
wget https://github.com/NetApp/trident/releases/download/v26.02.0/trident-
installer-26.02.0.tar.gz
tar -xf trident-installer-26.02.0.tar.gz
cd trident-installer
```

Étape 2 : Créez le `TridentOrchestrator` CRD

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

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

Étape 3 : Déployer l'opérateur Trident

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

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

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

Avant de commencer

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

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

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

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

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

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

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

Étapes

1. Créer les ressources et déployer l'opérateur :

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

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

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

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

Étape 4 : Créez le `TridentOrchestrator` et installez Trident

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

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

kubectl describe torc trident

Name:        trident
Namespace:
Labels:      <none>
Annotations: <none>
API Version: trident.netapp.io/v1
Kind:        TridentOrchestrator
...
Spec:
  Debug:     true
  Namespace: trident
  nodePrep:
  - iscsi
Status:
  Current Installation Params:
    IPv6:                      false
    Autosupport Hostname:
    Autosupport Image:         netapp/trident-autosupport:26.02
    Autosupport Proxy:
    Autosupport Serial Number:
    Debug:                     true
    Image Pull Secrets:
    Image Registry:
    k8sTimeout:           30
    Kubelet Dir:          /var/lib/kubelet
    Log Format:           text
    Silence Autosupport:  false
    Trident Image:        netapp/trident:26.02.0
  Message:                  Trident installed  Namespace:
trident
  Status:                   Installed
  Version:                  v26.02.0
Events:
    Type Reason Age From Message ---- ------ ---- ---- -------Normal
    Installing 74s trident-operator.netapp.io Installing Trident Normal
    Installed 67s trident-operator.netapp.io Trident installed
```

Vérifiez l’installation

Il existe plusieurs façons de vérifier votre installation.

Utilisation `TridentOrchestrator` du statut

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

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

Utilisation de l'état de création du pod

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

```
kubectl get pods -n trident

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

En utilisant `tridentctl`

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

```
./tridentctl -n trident version

+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 26.02.0        | 26.02.0        |
+----------------+----------------+
```

Déployez manuellement l'opérateur Trident (mode hors ligne)

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

Informations essentielles sur Trident

Vous devez lire les informations essentielles suivantes concernant Trident.

Informations essentielles sur Trident

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

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

Déployez manuellement l'opérateur Trident et installez Trident

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

Avant de commencer

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

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

1. Vérifiez votre version de Kubernetes :

```
kubectl version
```

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

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

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

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

Étape 1 : Téléchargez le package d'installation de Trident

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

```
wget https://github.com/NetApp/trident/releases/download/v6.0/trident-
installer-26.02.0.tar.gz
tar -xf trident-installer-26.02.0.tar.gz
cd trident-installer
```

Étape 2 : Créez le `TridentOrchestrator` CRD

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

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

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

Dans `/deploy/operator.yaml`, mettez à jour `image: docker.io/netapp/trident-operator:26.02.0` pour refléter l'emplacement de votre registre d'images. Votre "Images de Trident et de CSI" peut être situé dans un registre ou dans des registres différents, mais toutes les images CSI doivent être situées dans le même registre. Par exemple :

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

- `image: <your-registry>/netapp/trident-operator:26.02.0` si votre image Trident se trouve dans un registre différent de celui de vos images CSI.

Étape 4 : Déployer l'opérateur Trident

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

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

Avant de commencer

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

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

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

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

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

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

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

Étapes

1. Créer les ressources et déployer l'opérateur :

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

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

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

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

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

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

Images dans un seul registre

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

Images dans différents registres

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

Étape 6 : Créez le `TridentOrchestrator` et installez Trident

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

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

kubectl describe torc trident

Name:        trident
Namespace:
Labels:      <none>
Annotations: <none>
API Version: trident.netapp.io/v1
Kind:        TridentOrchestrator
...
Spec:
  Autosupport Image:  <your-registry>/trident-autosupport:26.02
  Debug:              true
  Image Registry:     <your-registry>
  Namespace:          trident
  Trident Image:      <your-registry>/trident:26.02.0
Status:
  Current Installation Params:
    IPv6:                       false
    Autosupport Hostname:
    Autosupport Image:          <your-registry>/trident-autosupport:26.02
    Autosupport Proxy:
    Autosupport Serial Number:
    Debug:                      true
    Http Request Timeout:       90s
    Image Pull Secrets:
    Image Registry:       <your-registry>
    k8sTimeout:           30
    Kubelet Dir:          /var/lib/kubelet
    Log Format:           text
    Probe Port:           17546
    Silence Autosupport:  false
    Trident Image:        <your-registry>/trident:26.02.0
  Message:                Trident installed
  Namespace:              trident
  Status:                 Installed
  Version:                v26.02.0
Events:
    Type Reason Age From Message ---- ------ ---- ---- -------Normal
    Installing 74s trident-operator.netapp.io Installing Trident Normal
    Installed 67s trident-operator.netapp.io Trident installed
```

Vérifiez l'installation

Il existe plusieurs façons de vérifier votre installation.

Utilisation `TridentOrchestrator` du statut

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

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

Utilisation de l'état de création du pod

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

```
kubectl get pods -n trident

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

En utilisant `tridentctl`

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

```
./tridentctl -n trident version

+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 26.02.0        | 26.02.0        |
+----------------+----------------+
```

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

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

Informations essentielles concernant Trident 25.10

Vous devez lire les informations essentielles suivantes concernant Trident.

Informations essentielles sur Trident

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

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

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

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

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

Avant de commencer

En plus de "prérequis de déploiement" vous avez besoin de "Helm version 3".

Étapes

1. Ajoutez le dépôt Trident Helm :

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

2. Utilisez `helm install` et spécifiez un nom pour votre déploiement comme dans l'exemple suivant où `100..0` est la version de Trident que vous installez.

```
helm install <name> netapp-trident/trident-operator --version 100.2602.0
--create-namespace --namespace <trident-namespace>
```

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

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

Transmettez les données de configuration lors de l'installation

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

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

Par exemple, pour modifier la valeur par défaut de `debug`, exécutez la commande suivante où `100.2602.0` est la version de Trident que vous installez :

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

Options de configuration

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

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

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

`affinity`	Affinité pour l'affectation des pods	`affinity:`
		`  nodeAffinity:`
		
		`requiredDuringSchedulingIgnoredDuringExecution:`
		`  nodeSelectorTerms:`
		`  - matchExpressions:`
		`  - key: kubernetes.io/arch`
		`  operator: In`
		`  values:`
		`  - arm64`
		`  - amd64`
		`  - key: kubernetes.io/os`
		`  operator: In`
		`  values:`
		`  - linux`
		
		Ne supprimez pas l'affinité par défaut du fichier values.yaml. Lorsque vous souhaitez fournir une affinité personnalisée, étendez l'affinité par défaut.

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

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

Option	Description	Défaut
`tridentK8sTimeout`	Remplace le délai d'expiration par défaut de 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"`
`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.	`""`

Option	Description	Défaut
`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.	`""`
`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`

Option	Description	Défaut
`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 les opérations simultanées du contrôleur Trident pour un débit amélioré. **Aperçu technique** : Cette fonctionnalité est expérimentale et prend actuellement en charge des flux de travail parallèles limités avec les pilotes ONTAP-NAS (NFS uniquement) et ONTAP-SAN (NVMe pour unified ONTAP 9), en plus de l'aperçu technique existant pour le pilote ONTAP-SAN (protocoles iSCSI et FCP dans unified ONTAP 9).	false
`k8sAPIQPS`	Limite de requêtes par seconde (QPS) utilisée par le contrôleur lors de la communication avec le serveur d'API Kubernetes. La valeur de Burst est définie automatiquement en fonction de la valeur QPS.	`100`; facultatif

Option	Description	Défaut

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

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

elles ne sont pas spécifiées.
- Les noms des conteneurs sont apparaissent dans les spécifications du pod.
- Les sidecars sont répertoriés sous chaque conteneur principal.
- Consultez le champ `status.CurrentInstallationParams` TORC pour afficher les valeurs actuellement appliquées.

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

Comprendre les pods de contrôleur et les pods de nœud

Trident s'exécute sous la forme d'un conteneur 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 "tétrances et taints" sont utilisés pour contraindre un pod à s'exécuter sur un nœud spécifique ou préféré. En utilisant le `ControllerPlugin` et `NodePlugin`, vous pouvez spécifier des contraintes et des substitutions.

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

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

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

Informations essentielles concernant Trident 26.02

Vous devez lire les informations essentielles suivantes concernant Trident.

Informations essentielles sur Trident

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

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

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

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

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

Avant de commencer

En plus de "prérequis de déploiement" vous avez besoin de "Helm version 3".

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

Étapes

1. Ajoutez le dépôt Trident Helm :

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

2. Utilisez `helm install` et spécifiez un nom pour votre déploiement et l'emplacement du registre d'images. Votre "Images de Trident et de CSI" peut être situé dans un registre ou dans des registres différents, mais toutes les images CSI doivent être situées dans le même registre. Dans les exemples, `100.2602.0` correspond à la version de Trident que vous installez.

Images dans un seul registre

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

Images dans différents registres

```
helm install <name> netapp-trident/trident-operator --version
100.2602.0 --set imageRegistry=<your-registry> --set operatorImage
=<your-registry>/trident-operator:26.02.0 --set
tridentAutosupportImage=<your-registry>/trident-autosupport:26.02
--set tridentImage=<your-registry>/trident:26.02.0 --create
-namespace --namespace <trident-namespace> --set nodePrep={iscsi}
```

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

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

Transmettez les données de configuration lors de l'installation

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

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

Par exemple, pour modifier la valeur par défaut de `debug`, exécutez la commande suivante où `100.2602.0` est la version de Trident que vous installez :

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

Pour ajouter la valeur nodePrep, exécutez la commande suivante :

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

Options de configuration

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

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

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

Option	Description	Défaut

`affinity`	Affinité pour l'affectation des pods	`affinity:`
		`  nodeAffinity:`
		
		`requiredDuringSchedul`
		`ingIgnoredDuringExecu`
		`tion:`
		
		`nodeSelectorTerms:`
		`  -`
		`matchExpressions:`
		`  - key:`
		`kubernetes.io/arch`
		
		`operator: In`
		`  values:`
		`  - arm64`
		`  - amd64`
		`  - key:`
		`kubernetes.io/os`
		
		`operator: In`
		`  values:`
		`  - linux`
		
		Ne supprimez pas l'affinité par défaut du fichier values.yaml. Lorsque vous souhaitez fournir une affinité personnalisée, étendez l'affinité par défaut.

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

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

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

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

Option	Description	Défaut

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

Personnaliser l'installation de l'opérateur Trident

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

Comprendre les pods de contrôleur et les pods de nœud

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

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

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

Options de configuration

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

Ce tableau détaille `TridentOrchestrator` les attributs.

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

- Aucune limite n'est appliquée par défaut - sauf les 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 `status.CurrentInstallationParams` TORC pour afficher les valeurs actuellement appliquées.

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

Paramètre	Description	Défaut
`cloudProvider`	Définissez sur `"Azure"` lors de l'utilisation d'identités gérées ou d'une identité cloud sur un cluster AKS. Définir sur `"AWS"` lors de l'utilisation d'une identité cloud sur un cluster EKS. Définir sur `"GCP"` lors de l'utilisation d'une identité cloud sur un cluster GKE.	`""`
`cloudIdentity`	Définissez sur workload identity (« azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxx ») lors de l'utilisation de l'identité cloud sur un cluster AKS. Définissez sur AWS IAM role (« 'eks.amazonaws.com/role-arn: arn:aws:iam::123456:role/trident-role' ») lors de l'utilisation de l'identité cloud sur un cluster EKS. Définissez sur cloud identity ("'iam.gke.io/gcp-service-account: xxxx@mygcpproject.iam.gserviceaccount.com'") lors de l'utilisation de l'identité cloud sur un cluster GKE.	`""`
`IPv6`	Installer Trident sur IPv6	false
`k8sTimeout`	Délai d'attente pour les opérations Kubernetes. Le `k8sTimeout`paramètre s'applique uniquement à l'installation de Trident.	`180sec`
`silenceAutosupport`	N'envoyez pas automatiquement les bundles d'autosupport à NetApp	`false`
`autosupportImage`	L'image conteneur pour la télémétrie Autosupport	`"netapp/trident-autosupport10"`
`autosupportProxy`	L'adresse/port d'un proxy pour l'envoi des données de télémétrie Autosupport	`"http://proxy.example.com:8888"`
`uninstall`	Un indicateur utilisé pour désinstaller Trident	`false`
`logFormat`	Format de journalisation Trident à utiliser [text,json]	`"text"`
`tridentImage`	Image Trident à installer	`"netapp/trident:26.02"`
`imageRegistry`	Chemin d'accès au registre interne, du format `<registry FQDN>[:port][/subpath]`	`"registry.k8s.io"`
`kubeletDir`	Chemin d'accès au répertoire kubelet sur l'hôte	`"/var/lib/kubelet"`
`wipeout`	Une liste des ressources à supprimer pour effectuer une suppression complète de Trident	
`imagePullSecrets`	Secrets pour extraire des images d'un registre interne	

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

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

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

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

Exemples de configurations

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

Configuration personnalisée de base

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

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

Sélecteurs de nœuds

Cet exemple installe Trident avec des sélecteurs de nœuds.

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

memory:
liveness-

Nœuds de travail Windows

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

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

Identités gérées sur un cluster AKS

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

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

Identité cloud sur un cluster AKS

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

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

Identité cloud sur un cluster EKS

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

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  cloudProvider: "AWS"
  cloudIdentity: "'eks.amazonaws.com/role-arn:
arn:aws:iam::123456:role/trident-role'"
```

Identité cloud pour GKE

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

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

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

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

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

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

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

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

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

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

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

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

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

Informations sur le copyright

Informations sur les marques commerciales