



Trident installieren

Trident

NetApp
July 01, 2026

Inhalt

Trident installieren	1
Erfahren Sie mehr über die Trident-Installation	1
Wichtige Informationen zu Trident 26.02	1
Bevor Sie beginnen	1
Wählen Sie Ihre Installationsmethode	2
Wählen Sie Ihren Installationsmodus	4
Wählen Sie den Prozess basierend auf Ihrer Methode und Ihrem Modus aus	4
Wechsel zwischen Installationsmethoden	5
Weitere bekannte Konfigurationsoptionen	5
Installation mit dem Trident operator	5
Manuelle Bereitstellung des Trident Operators (Standardmodus)	5
Manuelle Bereitstellung des Trident Operators (Offline-Modus)	11
Trident Operator mit Helm bereitstellen (Standardmodus)	17
Trident operator mit Helm bereitstellen (Offline mode)	26
Trident-Operator-Installation anpassen	35
Installation mit tridentctl	47
Installation mit tridentctl	47
tridentctl-Installation anpassen	51
Installation mit dem OpenShift zertifizierten Operator	52
Installieren Sie Trident mit OpenShift OperatorHub	52
Wechseln Sie zum OpenShift zertifizierten Trident Operator	55

Trident installieren

Erfahren Sie mehr über die Trident-Installation

Um sicherzustellen, dass Trident in einer Vielzahl von Umgebungen und Organisationen installiert werden kann, bietet NetApp mehrere Installationsoptionen an. Sie können Trident mithilfe des Trident-Operators (manuell oder mit Helm) oder mit `tridentctl` installieren. Dieses Thema enthält wichtige Informationen zur Auswahl des richtigen Installationsprozesses für Sie.

Wichtige Informationen zu Trident 26.02

Sie müssen die folgenden wichtigen Informationen über Trident lesen.

Wichtige Informationen zu Trident

- Kubernetes 1.35 wird nun in Trident unterstützt. Trident sollte vor Kubernetes aktualisiert werden.
- Trident setzt die Verwendung der Multipathing-Konfiguration in SAN-Umgebungen strikt durch, mit einem empfohlenen Wert von `find_multipaths: no` in der `multipath.conf`-Datei.

Die Verwendung einer Konfiguration ohne Multipathing oder die Verwendung von `find_multipaths: yes` oder `find_multipaths: smart` Wert in der Datei `multipath.conf` führt zu Mount-Fehlern. Trident hat die Verwendung von `find_multipaths: no` seit der Version 21.07 empfohlen.

Bevor Sie beginnen

Unabhängig vom Installationsweg müssen Sie Folgendes haben:

- Vollständige Berechtigungen für einen unterstützten Kubernetes-Cluster, auf dem eine unterstützte Kubernetes-Version ausgeführt wird und die erforderlichen Funktionen aktiviert sind. Überprüfen Sie die ["Anforderungen"](#) für Details.
- Zugriff auf ein unterstütztes NetApp Speichersystem.
- Fähigkeit, Volumes von allen Kubernetes-Worker-Knoten einzubinden.
- Ein Linux-Host mit `kubectl` (oder `oc`, wenn Sie OpenShift verwenden) installiert und konfiguriert, um den Kubernetes-Cluster zu verwalten, den Sie verwenden möchten.
- Die `KUBECONFIG` Umgebungsvariable, die auf Ihre Kubernetes-Clusterkonfiguration verweist.
- Wenn Sie Kubernetes mit Docker Enterprise verwenden ["Folgen Sie deren Schritten, um den CLI-Zugriff zu aktivieren"](#).
- Der Cluster muss privilegierte Workloads unterstützen.



Falls Sie sich noch nicht mit dem ["Grundkonzept"](#) vertraut gemacht haben, ist jetzt ein guter Zeitpunkt dafür.

Wählen Sie Ihre Installationsmethode

Wählen Sie die für Sie passende Installationsmethode. Sie sollten auch die Überlegungen zu ["Wechsel zwischen Methoden"](#) prüfen, bevor Sie Ihre Entscheidung treffen.

Verwendung des Trident-Operators

Ob manuell oder mit Helm – der Trident-Operator ist eine großartige Möglichkeit, die Installation zu vereinfachen und Trident-Ressourcen dynamisch zu verwalten. Sie können sogar ["Passen Sie Ihre Trident-Operator-Bereitstellung an"](#) unter Verwendung der Attribute in der `TridentOrchestrator` Custom Resource (CR).

Zu den Vorteilen der Verwendung des Trident Operators gehören:

Trident Objekterstellung

Der Trident Operator erstellt automatisch die folgenden Objekte für Ihre Kubernetes-Version.

- ServiceAccount für den Operator
- ClusterRole und ClusterRoleBinding zum ServiceAccount
- Dedizierte PodSecurityPolicy (für Kubernetes 1.25 und früher)
- Der Operator selbst

Ressourcenverantwortung

Der clusterweite Trident-Operator verwaltet Ressourcen einer Trident-Installation auf Clusterebene. Dies mindert Fehler, die bei der Verwaltung clusterweiter Ressourcen mit einem namensraumweiten Operator auftreten können. Dies ist für Selbstreparatur und das Einspielen von Patches unerlässlich.

Selbstreparatur-Fähigkeit

Der Operator überwacht die Trident-Installation und ergreift aktiv Maßnahmen, um Probleme zu beheben, beispielsweise wenn die Bereitstellung gelöscht oder versehentlich geändert wird. Ein `trident-operator-<generated-id>` Pod wird erstellt, der eine `TridentOrchestrator` CR mit einer Trident-Installation verknüpft. Dies stellt sicher, dass sich nur eine Instanz von Trident im Cluster befindet und deren Einrichtung kontrolliert wird, sodass die Installation idempotent ist. Wenn Änderungen an der Installation vorgenommen werden (wie das Löschen der Bereitstellung oder des Node-Daemonsets), erkennt der Operator diese und behebt sie einzeln.

Einfache Aktualisierungen bestehender Installationen

Sie können eine bestehende Bereitstellung mithilfe des Operators problemlos aktualisieren. Sie müssen lediglich die `TridentOrchestrator` CR bearbeiten, um Aktualisierungen an einer Installation vorzunehmen.

Nehmen Sie zum Beispiel ein Szenario, in dem Sie Trident so konfigurieren müssen, dass Debug-Protokolle generiert werden. Dazu patchen Sie Ihre `TridentOrchestrator` so, dass `spec.debug` auf `true` gesetzt wird:

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

Nachdem `TridentOrchestrator` aktualisiert wurde, verarbeitet der Operator die Updates und patcht die bestehende Installation. Dies kann die Erstellung neuer Pods auslösen, um die Installation entsprechend anzupassen.

Saubere Neuinstallation

Der clusterweite Trident-Operator ermöglicht die saubere Entfernung clusterweiter Ressourcen. Benutzer können Trident vollständig deinstallieren und problemlos neu installieren.

Automatische Kubernetes-Upgrade-Verarbeitung

Wenn die Kubernetes-Version des Clusters auf eine unterstützte Version aktualisiert wird, aktualisiert der Operator eine bestehende Trident-Installation automatisch und ändert sie, um sicherzustellen, dass sie den Anforderungen der Kubernetes-Version entspricht.



Wird der Cluster auf eine nicht unterstützte Version aktualisiert, verhindert der Operator die Installation von Trident. Falls Trident bereits mit dem Operator installiert wurde, wird eine Warnung angezeigt, die darauf hinweist, dass Trident auf einer nicht unterstützten Kubernetes-Version installiert ist.

Verwendung `tridentctl`

Wenn Sie eine bestehende Installation haben, die aktualisiert werden muss, oder wenn Sie Ihre Installation stark anpassen möchten, sollten Sie in Betracht ziehen. Dies ist die herkömmliche Methode zur Bereitstellung von Trident.

Sie können verwenden, um die Manifeste für Trident-Ressourcen zu generieren. Dies umfasst die Bereitstellung, das DaemonSet, das Dienstkonto und die Clusterrolle, die Trident im Rahmen der Installation erstellt.



Ab Version 22.04 werden AES-Schlüssel nicht mehr bei jeder Trident-Installation neu generiert. Mit dieser Version installiert Trident ein neues geheimes Objekt, das über zukünftige Installationen hinweg erhalten bleibt. Das bedeutet, `tridentctl` dass unter Version 22.04 frühere Versionen von Trident deinstalliert werden können, ältere Versionen jedoch nicht Installationen der Version 22.04 deinstallieren können. Wählen Sie die passende Installationsmethode.

Wählen Sie Ihren Installationsmodus

Bestimmen Sie Ihren Bereitstellungsprozess basierend auf dem von Ihrer Organisation benötigten *Installationsmodus* (Standard, Offline oder Remote).

Standardinstallation

Dies ist die einfachste Methode, Trident zu installieren, und funktioniert in den meisten Umgebungen, die keine Netzwerkbeschränkungen haben. Der Standardinstallationsmodus verwendet Standardregistries, um die erforderlichen Trident (`docker.io` und CSI (`registry.k8s.io` Images zu speichern.

Wenn Sie den Standardmodus verwenden, führt das Trident-Installationsprogramm Folgendes aus:

- Ruft die Container-Images über das Internet ab
- Erstellt ein Deployment oder einen Node-Daemonset, der Trident Pods auf allen geeigneten Nodes im Kubernetes-Cluster startet

Offline-Installation

Der Offline-Installationsmodus kann in einer vom Netzwerk getrennten oder gesicherten Umgebung erforderlich sein. In diesem Szenario können Sie eine einzelne private, gespiegelte Registry oder zwei gespiegelte Registries erstellen, um die benötigten Trident- und CSI-Images zu speichern.



Unabhängig von Ihrer Registry-Konfiguration müssen CSI-Images in einer einzigen Registry vorhanden sein.

Ferninstallation

Hier ist ein Überblick auf hoher Ebene über den Ablauf der Ferninstallation:

- Installieren Sie die entsprechende Version von `kubectl` auf dem Remote-Rechner, von dem aus Sie Trident bereitstellen möchten.
- Kopieren Sie die Konfigurationsdateien aus dem Kubernetes-Cluster und setzen Sie die `KUBECONFIG` Umgebungsvariable auf dem Remote-Rechner.
- Initiieren Sie einen `kubectl get nodes`-Befehl, um zu überprüfen, ob Sie eine Verbindung zum erforderlichen Kubernetes-Cluster herstellen können.
- Schließen Sie die Bereitstellung von der Remote-Maschine aus mit den Standardinstallationsschritten ab.

Wählen Sie den Prozess basierend auf Ihrer Methode und Ihrem Modus aus

Nachdem Sie Ihre Entscheidungen getroffen haben, wählen Sie den entsprechenden Prozess aus.

Verfahren	Installationsmodus
Trident operator (manuell)	"Standardinstallation" "Offline-Installation"
Trident-Operator (Helm)	"Standardinstallation" "Offline-Installation"
tridentctl	"Standard- oder Offline-Installation"

Wechsel zwischen Installationsmethoden

Sie können Ihre Installationsmethode ändern. Beachten Sie vorher Folgendes:

- Verwenden Sie immer die gleiche Methode für die Installation und Deinstallation von Trident. Wenn Sie mit `tridentctl` bereitgestellt haben, sollten Sie die entsprechende Version der `tridentctl`-Binary verwenden, um Trident zu deinstallieren. Ebenso sollten Sie, wenn Sie mit dem Operator bereitstellen, die `TridentOrchestrator` CR bearbeiten und `spec.uninstall=true` setzen, um Trident zu deinstallieren.
- Wenn Sie eine operatorbasierte Bereitstellung haben, die Sie entfernen und stattdessen `tridentctl` zur Bereitstellung von Trident verwenden möchten, sollten Sie zuerst `TridentOrchestrator` bearbeiten und `spec.uninstall=true` setzen, um Trident zu deinstallieren. Löschen Sie dann `TridentOrchestrator` und die Operator-Bereitstellung. Sie können dann mit `tridentctl` installieren.
- Wenn Sie eine manuelle, operatorbasierte Bereitstellung haben und auf die Helm-basierte Trident Operator-Bereitstellung umsteigen möchten, sollten Sie den Operator zuerst manuell deinstallieren und dann die Helm-Installation durchführen. Dadurch kann Helm den Trident Operator mit den erforderlichen Labels und Annotationen bereitstellen. Wenn Sie dies nicht tun, schlägt Ihre Helm-basierte Trident Operator-Bereitstellung mit einem Label-Validierungsfehler und einem Annotations-Validierungsfehler fehl.
- Wenn Sie eine `tridentctl`-basierte Bereitstellung haben, können Sie eine Helm-basierte oder Operator-basierte Bereitstellung durchführen, ohne Trident zu deinstallieren.

Weitere bekannte Konfigurationsoptionen

Bei der Installation von Trident auf VMWare Tanzu Portfolio-Produkten:

- Das `--kubelet-dir`-Flag sollte auf den Speicherort des kubelet-Verzeichnisses gesetzt werden. Standardmäßig ist dies `/var/vcap/data/kubelet`.

Die Angabe des kubelet-Speicherorts mit `--kubelet-dir` ist bekannt dafür, bei Trident Operator-, Helm- und `tridentctl`-Deployments zu funktionieren.

Installation mit dem Trident operator

Manuelle Bereitstellung des Trident Operators (Standardmodus)

Sie können den Trident-Operator manuell bereitstellen, um Trident zu installieren. Dieser

Prozess gilt für Installationen, bei denen die von Trident benötigten Container-Images nicht in einer privaten Registry gespeichert sind. Wenn Sie eine private Image-Registry haben, verwenden Sie die "[Prozess für die Offline-Bereitstellung](#)".

Wichtige Informationen zu Trident 26.02

Sie müssen die folgenden wichtigen Informationen über Trident lesen.

Wichtige Informationen zu Trident

- Kubernetes 1.35 wird nun in Trident unterstützt. Trident sollte vor Kubernetes aktualisiert werden.
- Trident setzt die Verwendung der Multipathing-Konfiguration in SAN-Umgebungen strikt durch, mit einem empfohlenen Wert von `find_multipaths: no` in der `multipath.conf`-Datei.

Die Verwendung einer Konfiguration ohne Multipathing oder die Verwendung von `find_multipaths: yes` oder `find_multipaths: smart` Wert in der Datei `multipath.conf` führt zu Mount-Fehlern. Trident hat die Verwendung von `find_multipaths: no` seit der Version 21.07 empfohlen.

Den Trident-Operator manuell bereitstellen und Trident installieren

Überprüfen Sie "[die Installationsübersicht](#)", um sicherzustellen, dass Sie die Installationsvoraussetzungen erfüllt und die richtige Installationsoption für Ihre Umgebung ausgewählt haben.

Bevor Sie beginnen

Bevor Sie mit der Installation beginnen, melden Sie sich am Linux-Host an und vergewissern Sie sich, dass dieser ein funktionierendes "[unterstützter Kubernetes-Cluster](#)" verwaltet und dass Sie über die erforderlichen Berechtigungen verfügen.



Mit OpenShift verwenden Sie `oc` statt `kubectl` in allen folgenden Beispielen und melden Sie sich zuerst als **system:admin** an, indem Sie `oc login -u system:admin` oder `oc login -u kube-admin` ausführen.

1. Überprüfen Sie Ihre Kubernetes-Version:

```
kubectl version
```

2. Überprüfen Sie die Cluster-Administratorrechte:

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

3. Überprüfen Sie, ob Sie einen Pod starten können, der ein Image von Docker Hub verwendet, und Ihr Speichersystem über das Pod-Netzwerk erreichen können.

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

Schritt 1: Laden Sie das Trident-Installationspaket herunter

Das Trident-Installationspaket enthält alles, was Sie zum Bereitstellen des Trident-Operators und zum Installieren von Trident benötigen. Laden Sie die neueste Version des Trident-Installationsprogramms von ["der Abschnitt Assets auf GitHub"](#) herunter und extrahieren Sie sie.

```
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
```

Schritt 2: Erstellen Sie die TridentOrchestrator CRD

Erstellen Sie die `TridentOrchestrator` benutzerdefinierte Ressourcendefinition (CRD). Sie werden später eine `TridentOrchestrator` benutzerdefinierte Ressource erstellen. Verwenden Sie die entsprechende CRD-YAML-Version in `deploy/crds`, um die `TridentOrchestrator` CRD zu erstellen.

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

Schritt 3: Den Trident Operator einsetzen

Das Trident-Installationsprogramm stellt eine Bundle-Datei bereit, mit der der Operator installiert und zugehörige Objekte erstellt werden können. Die Bundle-Datei ist eine einfache Möglichkeit, den Operator bereitzustellen und Trident mit einer Standardkonfiguration zu installieren.

- Für Cluster, auf denen Kubernetes 1.24 läuft, verwenden Sie `bundle_pre_1_25.yaml`.

- Für Cluster, auf denen Kubernetes 1.25 oder höher läuft, verwenden Sie `bundle_post_1_25.yaml`.

Bevor Sie beginnen

- Standardmäßig installiert das Trident-Installationsprogramm den Operator im `trident` Namespace. Falls der `trident` Namespace nicht existiert, erstellen Sie ihn mit:

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

- Um den Operator in einem anderen Namespace als dem `trident` Namespace bereitzustellen, aktualisieren Sie `serviceaccount.yaml`, `clusterrolebinding.yaml` und `operator.yaml` und generieren Sie Ihre Bundle-Datei mit dem `kustomization.yaml`.

- a. Erstellen Sie das `kustomization.yaml` mit dem folgenden Befehl, wobei `<bundle.yaml>` `bundle_pre_1_25.yaml` oder `bundle_post_1_25.yaml` abhängig von Ihrer Kubernetes-Version ist.

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

- b. Kompilieren Sie das Bundle mit dem folgenden Befehl, wobei `<bundle.yaml>` `bundle_pre_1_25.yaml` oder `bundle_post_1_25.yaml` entsprechend Ihrer Kubernetes-Version ist.

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

Schritte

1. Erstellen Sie die Ressourcen und stellen Sie den Operator bereit:

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

2. Überprüfen Sie, ob der Operator, die Bereitstellung und die ReplicaSets erstellt wurden.

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



Es sollte nur **eine Instanz** des Operators in einem Kubernetes-Cluster geben. Erstellen Sie nicht mehrere Deployments des Trident Operators.

Schritt 4: Erstellen Sie das `TridentOrchestrator` und installieren Sie Trident

Sie können jetzt die `TridentOrchestrator` erstellen und Trident installieren. Optional können Sie ["Passen Sie Ihre Trident Installation individuell an"](#) mit den Attributen in der `TridentOrchestrator`-Spezifikation arbeiten.

```

kubect1 create -f deploy/crds/tridentorchestrator_cr.yaml
tridentorchestrator.trident.netapp.io/trident created

kubect1 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 ---- -
  Installing 74s trident-operator.netapp.io Installing Trident Normal
  Installed 67s trident-operator.netapp.io Trident installed

```

Überprüfen Sie die Installation

Es gibt mehrere Möglichkeiten, Ihre Installation zu überprüfen.

Status mit `TridentOrchestrator` verwenden

Der Status von `TridentOrchestrator` zeigt an, ob die Installation erfolgreich war und zeigt die installierte Version von Trident an. Während der Installation ändert sich der Status von `TridentOrchestrator` `Installing` zu `Installed`. Wenn Sie den `Failed` Status beobachten und der Operator das Problem nicht selbst beheben kann, "[Überprüfen Sie die Protokolle](#)".

Status	Beschreibung
Installation	Der Operator installiert Trident mit diesem <code>TridentOrchestrator</code> CR.
Installiert	Trident wurde erfolgreich installiert.
Deinstallation	Der Betreiber deinstalliert Trident, weil <code>spec.uninstall=true</code> .
Deinstalliert	Trident ist deinstalliert.
Fehlgeschlagen	Der Operator konnte Trident nicht installieren, patchen, aktualisieren oder deinstallieren; der Operator wird automatisch versuchen, sich von diesem Zustand zu erholen. Wenn dieser Zustand weiterhin besteht, ist eine Fehlerbehebung erforderlich.
Aktualisierung	Der Betreiber aktualisiert eine bestehende Installation.
Fehler	Das <code>TridentOrchestrator</code> wird nicht verwendet. Ein anderer existiert bereits.

Verwendung des Pod-Erstellungsstatus

Sie können bestätigen, ob die Trident Installation abgeschlossen wurde, indem Sie den Status der erstellten Pods überprüfen:

```
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
```

Verwendung `tridentctl`

Sie können `tridentctl` verwenden, um die installierte Version von Trident zu überprüfen.

```
./tridentctl -n trident version

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

Manuelle Bereitstellung des Trident Operators (Offline-Modus)

Sie können den Trident-Operator manuell bereitstellen, um Trident zu installieren. Dieser Vorgang gilt für Installationen, bei denen die von Trident benötigten Container-Images in einer privaten Registry gespeichert sind. Wenn Sie keine private Image-Registry besitzen, verwenden Sie die ["Prozess für die Standardbereitstellung"](#).

Wichtige Informationen zu Trident

Sie müssen die folgenden wichtigen Informationen über Trident lesen.

Wichtige Informationen zu Trident

- Kubernetes 1.35 wird nun in Trident unterstützt. Trident sollte vor Kubernetes aktualisiert werden.
- Trident setzt die Verwendung der Multipathing-Konfiguration in SAN-Umgebungen strikt durch, mit einem empfohlenen Wert von `find_multipaths: no` in der `multipath.conf`-Datei.

Die Verwendung einer Konfiguration ohne Multipathing oder die Verwendung von `find_multipaths: yes` oder `find_multipaths: smart` Wert in der Datei `multipath.conf` führt zu Mount-Fehlern. Trident hat die Verwendung von `find_multipaths: no` seit der Version 21.07 empfohlen.

Den Trident-Operator manuell bereitstellen und Trident installieren

Überprüfen Sie ["die Installationsübersicht"](#), um sicherzustellen, dass Sie die Installationsvoraussetzungen erfüllt und die richtige Installationsoption für Ihre Umgebung ausgewählt haben.

Bevor Sie beginnen

Melden Sie sich am Linux-Host an und vergewissern Sie sich, dass er ein funktionierendes und ["unterstützter Kubernetes-Cluster"](#) verwaltet und dass Sie über die erforderlichen Berechtigungen verfügen.



Mit OpenShift verwenden Sie `oc` statt `kubectl` in allen folgenden Beispielen und melden Sie sich zuerst als **system:admin** an, indem Sie `oc login -u system:admin` oder `oc login -u kube-admin` ausführen.

1. Überprüfen Sie Ihre Kubernetes-Version:

```
kubectl version
```

2. Überprüfen Sie die Cluster-Administratorrechte:

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

3. Überprüfen Sie, ob Sie einen Pod starten können, der ein Image von Docker Hub verwendet, und Ihr Speichersystem über das Pod-Netzwerk erreichen können.

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

Schritt 1: Laden Sie das Trident-Installationspaket herunter

Das Trident-Installationspaket enthält alles, was Sie zum Bereitstellen des Trident-Operators und zum Installieren von Trident benötigen. Laden Sie die neueste Version des Trident-Installationsprogramms von "[der Abschnitt Assets auf GitHub](#)" herunter und extrahieren Sie sie.

```
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
```

Schritt 2: Erstellen Sie die `TridentOrchestrator` CRD

Erstellen Sie die `TridentOrchestrator` benutzerdefinierte Ressourcendefinition (CRD). Sie werden später eine `TridentOrchestrator` benutzerdefinierte Ressource erstellen. Verwenden Sie die entsprechende CRD-YAML-Version in `deploy/crds`, um die `TridentOrchestrator` CRD zu erstellen:

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

Schritt 3: Aktualisieren Sie den Registrierungsspeicherort im Operator

In `/deploy/operator.yaml` `image: docker.io/netapp/trident-operator:26.02.0` aktualisieren Sie den Pfad, um den Speicherort Ihrer Image-Registry widerzuspiegeln. Ihre "[Trident und CSI images](#)" können sich in einer

Registry oder in verschiedenen Registrys befinden, aber alle CSI-Images müssen sich in derselben Registry befinden. Beispiel:

- `image: <your-registry>/trident-operator:26.02.0` wenn sich alle Ihre Images in einer einzigen Registry befinden.
- `image: <your-registry>/netapp/trident-operator:26.02.0` wenn sich Ihr Trident-Image in einem anderen Registry als Ihre CSI-Images befindet.

Schritt 4: Den Trident Operator einsetzen

Das Trident-Installationsprogramm stellt eine Bundle-Datei bereit, mit der der Operator installiert und zugehörige Objekte erstellt werden können. Die Bundle-Datei ist eine einfache Möglichkeit, den Operator bereitzustellen und Trident mit einer Standardkonfiguration zu installieren.

- Für Cluster, auf denen Kubernetes 1.24 läuft, verwenden Sie `bundle_pre_1_25.yaml`.
- Für Cluster, auf denen Kubernetes 1.25 oder höher läuft, verwenden Sie `bundle_post_1_25.yaml`.

Bevor Sie beginnen

- Standardmäßig installiert das Trident-Installationsprogramm den Operator im `trident` Namespace. Falls der `trident` Namespace nicht existiert, erstellen Sie ihn mit:

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

- Um den Operator in einem anderen Namespace als dem `trident` Namespace bereitzustellen, aktualisieren Sie `serviceaccount.yaml`, `clusterrolebinding.yaml` und `operator.yaml` und generieren Sie Ihre Bundle-Datei mit dem `kustomization.yaml`.
 - a. Erstellen Sie das `kustomization.yaml` mit dem folgenden Befehl, wobei `<bundle.yaml>` `bundle_pre_1_25.yaml` oder `bundle_post_1_25.yaml` abhängig von Ihrer Kubernetes-Version ist.

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

- b. Kompilieren Sie das Bundle mit dem folgenden Befehl, wobei `<bundle.yaml>` `bundle_pre_1_25.yaml` oder `bundle_post_1_25.yaml` entsprechend Ihrer Kubernetes-Version ist.

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

Schritte

1. Erstellen Sie die Ressourcen und stellen Sie den Operator bereit:

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

2. Überprüfen Sie, ob der Operator, die Bereitstellung und die ReplicaSets erstellt wurden.

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



Es sollte nur **eine Instanz** des Operators in einem Kubernetes-Cluster geben. Erstellen Sie nicht mehrere Deployments des Trident Operators.

Schritt 5: Aktualisieren Sie den Speicherort der Bildregistrierung in der `TridentOrchestrator`

Ihre "[Trident und CSI images](#)" können sich in einer Registry oder in verschiedenen Registries befinden, aber alle CSI-Images müssen sich in derselben Registry befinden. Aktualisieren Sie `deploy/crds/tridentorchestrator_cr.yaml`, um die zusätzlichen Speicherortangaben entsprechend Ihrer Registry-Konfiguration hinzuzufügen.

Images in einem Registry

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

Images in verschiedenen Registern

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

Schritt 6: Erstellen Sie das `TridentOrchestrator` und installieren Sie Trident

Sie können jetzt die `TridentOrchestrator` erstellen und Trident installieren. Optional können Sie "[Passen Sie Ihre Trident Installation individuell an](#)" mithilfe der Attribute in der `TridentOrchestrator`-Spezifikation weiter anpassen. Das folgende Beispiel zeigt eine Installation, bei der sich Trident- und CSI-Images in unterschiedlichen Registries befinden.

```

kubect1 create -f deploy/crds/tridentorchestrator_cr.yaml
tridentorchestrator.trident.netapp.io/trident created

kubect1 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

```

Überprüfen Sie die Installation

Es gibt mehrere Möglichkeiten, Ihre Installation zu überprüfen.

Status mit `TridentOrchestrator` verwenden

Der Status von `TridentOrchestrator` zeigt an, ob die Installation erfolgreich war und zeigt die installierte Version von Trident an. Während der Installation ändert sich der Status von `TridentOrchestrator` `Installing` zu `Installed`. Wenn Sie den `Failed` Status beobachten und der Operator das Problem nicht selbst beheben kann, "[Überprüfen Sie die Protokolle](#)".

Status	Beschreibung
Installation	Der Operator installiert Trident mit diesem <code>TridentOrchestrator</code> CR.
Installiert	Trident wurde erfolgreich installiert.
Deinstallation	Der Betreiber deinstalliert Trident, weil <code>spec.uninstall=true</code> .
Deinstalliert	Trident ist deinstalliert.
Fehlgeschlagen	Der Operator konnte Trident nicht installieren, patchen, aktualisieren oder deinstallieren; der Operator wird automatisch versuchen, sich von diesem Zustand zu erholen. Wenn dieser Zustand weiterhin besteht, ist eine Fehlerbehebung erforderlich.
Aktualisierung	Der Betreiber aktualisiert eine bestehende Installation.
Fehler	Das <code>TridentOrchestrator</code> wird nicht verwendet. Ein anderer existiert bereits.

Verwendung des Pod-Erstellungsstatus

Sie können bestätigen, ob die Trident Installation abgeschlossen wurde, indem Sie den Status der erstellten Pods überprüfen:

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

NAME	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

Verwendung tridentctl

Sie können `tridentctl` verwenden, um die installierte Version von Trident zu überprüfen.

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

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

Trident Operator mit Helm bereitstellen (Standardmodus)

Sie können den Trident-Operator bereitstellen und Trident mithilfe von Helm installieren. Dieser Prozess gilt für Installationen, bei denen die von Trident benötigten Container-Images nicht in einer privaten Registry gespeichert sind. Wenn Sie eine private Image-Registry haben, verwenden Sie die ["Prozess für die Offline-Bereitstellung"](#).

Wichtige Informationen zu Trident 25.10

Sie müssen die folgenden wichtigen Informationen über Trident lesen.

Wichtige Informationen zu Trident

- Kubernetes 1.35 wird nun in Trident unterstützt. Trident sollte vor Kubernetes aktualisiert werden.
- Trident setzt die Verwendung der Multipathing-Konfiguration in SAN-Umgebungen strikt durch, mit einem empfohlenen Wert von `find_multipaths: no` in der `multipath.conf`-Datei.

Die Verwendung einer Konfiguration ohne Multipathing oder die Verwendung von `find_multipaths: yes` oder `find_multipaths: smart` Wert in der Datei `multipath.conf` führt zu Mount-Fehlern. Trident hat die Verwendung von `find_multipaths: no` seit der Version 21.07 empfohlen.

Den Trident-Operator bereitstellen und Trident mit Helm installieren

Mit dem Trident "[Helm Chart](#)" können Sie den Trident Operator bereitstellen und Trident in einem Schritt installieren.

Überprüfen Sie "[die Installationsübersicht](#)", um sicherzustellen, dass Sie die Installationsvoraussetzungen erfüllt und die richtige Installationsoption für Ihre Umgebung ausgewählt haben.

Bevor Sie beginnen

Zusätzlich zu dem "[Bereitstellungsvoraussetzungen](#)" benötigen Sie "[Helm Version 3](#)".

Schritte

1. Fügen Sie das Trident Helm repository hinzu:

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

2. Verwenden Sie `helm install` und geben Sie einen Namen für Ihre Bereitstellung an, wie im folgenden Beispiel, wobei `100.0.0` die Version von Trident ist, die Sie installieren.

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



Wenn Sie bereits einen Namespace für Trident erstellt haben, wird durch den `--create-namespace` Parameter kein zusätzlicher Namespace erstellt.

Sie können `helm list` verwenden, um Installationsdetails wie Name, Namespace, Chart, Status, App-Version und Revisionsnummer zu überprüfen.

Konfigurationsdaten während der Installation übergeben

Es gibt zwei Möglichkeiten, Konfigurationsdaten während der Installation zu übergeben:

Option	Beschreibung
<code>--values (oder -f)</code>	Geben Sie eine YAML-Datei mit Überschreibungen an. Dies kann mehrfach angegeben werden, und die ganz rechts stehende Datei hat Vorrang.
<code>--set</code>	Geben Sie Überschreibungen in der Befehlszeile an.


Um beispielsweise den Standardwert von `debug` zu ändern, führen Sie den folgenden Befehl aus, wobei `100.2602.0` die Version von Trident ist, die Sie installieren:

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

Konfigurationsoptionen

Diese Tabelle und die `values.yaml` Datei, die Teil des Helm-Charts ist, enthalten die Liste der Schlüssel und ihrer Standardwerte.


Option	Beschreibung	Standard
<code>nodeSelector</code>	Knotenbezeichnungen für die Pod-Zuweisung	
<code>podAnnotations</code>	Pod-Anmerkungen	
<code>deploymentAnnotations</code>	Bereitstellungsanmerkungen	
<code>tolerations</code>	Toleranzen für die Pod-Zuordnung	


Option	Beschreibung	Standard
affinity	Affinität für die Pod-Zuweisung	<pre data-bbox="820 157 1481 940"> affinity: nodeAffinity: requiredDuringSchedulingIgnoredDuringExecution: nodeSelectorTerms: - matchExpressions: - key: kubernetes.io/arch operator: In values: - arm64 - amd64 - key: kubernetes.io/os operator: In values: - linux </pre> <div data-bbox="847 1045 906 1096" style="display: inline-block; vertical-align: middle;">  </div> <div data-bbox="966 982 1448 1159" style="display: inline-block; vertical-align: middle; margin-left: 10px;"> <p>Entfernen Sie die Standardaffinität nicht aus der Datei values.yaml. Wenn Sie eine benutzerdefinierte Affinität angeben möchten, erweitern Sie die Standardaffinität.</p> </div>
tridentControllerPluginNodeSelector	Zusätzliche Knotenselektoren für Pods. Siehe Controller-Pods und Node-Pods verstehen für Details.	
tridentControllerPluginTolerations	Überschreibt die Kubernetes-Toleranzen für Pods. Siehe Controller-Pods und Node-Pods verstehen für Details.	
tridentNodePluginNodeSelector	Zusätzliche Knotenselektoren für Pods. Siehe Controller-Pods und Node-Pods verstehen für Details.	
tridentNodePluginTolerations	Überschreibt die Kubernetes-Toleranzen für Pods. Siehe Controller-Pods und Node-Pods verstehen für Details.	

Option	Beschreibung	Standard
imageRegistry	Identifiziert die Registry für die <code>trident-operator</code> , <code>trident</code> und andere Images. Leer lassen, um den Standardwert zu akzeptieren. WICHTIG: Wenn Sie Trident in einem privaten Repository installieren und den <code>imageRegistry</code> -Schalter zur Angabe des Repository-Speicherorts verwenden, verwenden Sie <code>/netapp/</code> nicht im Repository-Pfad.	""
imagePullPolicy	Legt die Image-Pull-Richtlinie für das <code>trident-operator</code> fest.	IfNotPresent
imagePullSecrets	Legt die Image-Pull-Secrets für die <code>trident-operator</code> , <code>trident</code> und andere Images fest.	
kubeletDir	Ermöglicht das Überschreiben des Host-Speicherorts des internen Zustands von kubelet.	"/var/lib/kubelet"
operatorLogLevel	Ermöglicht das Protokollierungslevel des Trident-Operators auf: <code>trace</code> , <code>debug</code> , <code>info</code> , <code>warn</code> , <code>error</code> oder <code>fatal</code> festzulegen.	"info"
operatorDebug	Ermöglicht das Festlegen des Protokollierungsgrads des Trident-Operators auf Debug.	true
operatorImage	Ermöglicht das vollständige Überschreiben des Bildes für <code>trident-operator</code> .	""
operatorImageTag	Ermöglicht das Überschreiben des Tags des <code>trident-operator</code> -Images.	""
tridentIPv6	Ermöglicht die Aktivierung von Trident für den Einsatz in IPv6-Clustern.	false
tridentK8sTimeout	Überschreibt die standardmäßige 30-Sekunden-Zeitüberschreitung für die meisten Kubernetes-API-Operationen (falls ungleich Null, in Sekunden).	0

Option	Beschreibung	Standard
<code>tridentHttpRequestTimeout</code>	Überschreibt die standardmäßige 90-Sekunden-Zeitüberschreitung für HTTP-Anfragen, wobei 0s eine unbegrenzte Dauer für die Zeitüberschreitung bedeutet. Negative Werte sind nicht zulässig.	"90s"
<code>tridentSilenceAutoSupport</code>	Ermöglicht das Deaktivieren der periodischen AutoSupport-Berichterstattung von Trident.	false
<code>tridentAutoSupportImageTag</code>	Ermöglicht das Überschreiben des Tags des Images für den Trident AutoSupport-Container.	<version>
<code>tridentAutoSupportProxy</code>	Ermöglicht dem Trident AutoSupport Container, über einen HTTP-Proxy nach Hause zu telefonieren.	""
<code>tridentLogFormat</code>	Legt das Trident Protokollierungsformat (<code>text</code> oder <code>json</code>) fest.	"text"
<code>tridentDisableAuditLog</code>	Deaktiviert den Trident Audit-Logger.	true
<code>tridentLogLevel</code>	Ermöglicht das Protokollierungslevel von Trident auf <code>trace</code> , <code>debug</code> , <code>info</code> , <code>warn</code> , <code>error</code> oder <code>fatal</code> festzulegen.	"info"
<code>tridentDebug</code>	Ermöglicht das Festlegen des Protokollierungslevels von Trident auf <code>debug</code> . Sie können den erzwungenen Trennvorgang durch Integration mit dem <code>node health check (NHC) Operator</code> automatisieren. Weitere Informationen finden Sie unter "Automatisierung des Failovers von zustandsbehafteten Anwendungen mit Trident" .	false
<code>tridentLogWorkflows</code>	Ermöglicht die Aktivierung bestimmter Trident-Workflows für die Protokollierung von Ablaufverfolgungen oder die Unterdrückung von Protokollen.	""
<code>tridentLogLayers</code>	Ermöglicht die Aktivierung bestimmter Trident-Ebenen für die Protokollierung von Ablaufverfolgungen oder die Unterdrückung von Protokollen.	""

Option	Beschreibung	Standard
tridentImage	Ermöglicht das vollständige Überschreiben des Images für Trident.	""
tridentImageTag	Ermöglicht das Überschreiben des Tags des Images für Trident.	""
tridentProbePort	Ermöglicht das Überschreiben des Standardports, der für Kubernetes Liveness-/Readiness-Probes verwendet wird.	""
windows	Ermöglicht die Installation von Trident auf einem Windows-Worker-Knoten.	false
enableForceDetach	Ermöglicht das Aktivieren der Funktion „Force Detach“.	false
excludePodSecurityPolicy	Schließt die Sicherheitsrichtlinie für den Operator-Pod von der Erstellung aus.	false
cloudProvider	Auf "Azure" setzen, wenn verwaltete Identitäten oder eine Cloud-Identität auf einem AKS-Cluster verwendet werden. Auf "AWS" setzen, wenn eine Cloud-Identität auf einem EKS-Cluster verwendet wird.	""
cloudIdentity	Legen Sie auf die Workload-Identität („azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxx“) fest, wenn Sie Cloud Identity auf einem AKS-Cluster verwenden. Legen Sie auf die AWS IAM-Rolle („eks.amazonaws.com/role-arn: arn:aws:iam::123456:role/trident-role“) fest, wenn Sie Cloud Identity auf einem EKS-Cluster verwenden.	""
iscsiSelfHealingInterval	Das Intervall, in dem die iSCSI Selbstreparatur aufgerufen wird.	5m0s
iscsiSelfHealingWaitTime	Die Zeitspanne, nach der die iSCSI Selbstreparatur einen Versuch unternimmt, eine veraltete Sitzung durch Durchführung einer Abmeldung und anschließenden Anmeldung zu beheben.	7m0s

Option	Beschreibung	Standard
nodePrep	Ermöglicht Trident, die Knoten des Kubernetes-Clusters für die Verwaltung von Volumes mithilfe des angegebenen Datenspeicherprotokolls vorzubereiten. Derzeit ist <code>iscsi</code> der einzige unterstützte Wert. HINWEIS: Ab OpenShift 4.19 ist die Mindestversion von Trident für diese Funktion 25.06.1.	
enableConcurrency	Ermöglicht gleichzeitige Trident-Controller-Operationen für einen verbesserten Durchsatz. <div style="border: 1px solid #ccc; padding: 10px; margin: 10px 0;">  <p>Tech Preview: Diese Funktion ist experimentell und unterstützt derzeit eingeschränkte parallele Arbeitsabläufe mit den ONTAP-NAS (nur NFS) und ONTAP-SAN (NVMe für unified ONTAP 9) Treibern, zusätzlich zur bestehenden Tech Preview für den ONTAP-SAN Treiber (iSCSI- und FCP-Protokolle in unified ONTAP 9).</p> </div>	false
k8sAPIQPS	Die vom Controller bei der Kommunikation mit dem Kubernetes-API-Server verwendete Abfrageanzahl pro Sekunde (QPS). Der Burst-Wert wird automatisch anhand des QPS-Werts festgelegt.	100; optional

Option	Beschreibung	Standard
resources	<p>Legt Kubernetes-Ressourcenlimits und -anforderungen für die Trident-Controller-, Node- und Operator-Pods fest. Sie können CPU und Arbeitsspeicher für jeden Container und Sidecar konfigurieren, um die Ressourcenzuweisung in Kubernetes zu verwalten.</p> <p>Weitere Informationen zur Konfiguration von Ressourcenanforderungen und -limits finden Sie unter "Ressourcenverwaltung für Pods und Container".</p> <div style="border-left: 1px solid #ccc; padding-left: 10px; margin-left: 20px;">  <ul style="list-style-type: none"> • Ändern Sie NICHT die Namen von Containern oder Feldern. • Ändern Sie die Einrückung NICHT – die YAML-Einrückung ist für das korrekte Parsen entscheidend. </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: linux: trident-main:</pre>

Option	Beschreibung	Standard
httpsMetrics	Aktivieren Sie HTTPS für den Prometheus metrics endpoint.	false
hostNetwork	Aktiviert die Host-Netzwerkverbindung für den Trident Controller. Dies ist nützlich, wenn Sie den Frontend- und Backend-Datenverkehr in einem Multi-Home-Netzwerk trennen möchten.	false

wenn sie nicht angegeben sind.

Controller-Pods und Node-Pods verstehen

- Containernamen werden so angegeben, wie sie in den Pod-Spezifikationen erscheinen.

Trident läuft als einzelner Controller-Pod sowie als Node-Pod auf jedem Worker-Knoten im Cluster. Der Node-Pod muss auf jedem Host ausgeführt werden, auf dem Sie möglicherweise ein Trident Volume einbinden möchten.

 Kubernetes **"Knotenselektoren"** und **"Tolerations und Taints"** werden verwendet, um einen Pod auf einen bestimmten oder bevorzugten Knoten zu beschränken. Mit dem Controller-Plugin und NodePlugin können Sie Einschränkungen und Überschreibungen festlegen.

- Das Controller-Plugin übernimmt die Bereitstellung und Verwaltung von Volumes, wie Snapshots und Größenänderungen.
- Das Node-Plugin übernimmt das Anbinden des Speichers an den Knoten.

Trident operator mit Helm bereitstellen (Offline mode)

Sie können den Trident-Operator bereitstellen und Trident mithilfe von Helm installieren. Dieser Vorgang gilt für Installationen, bei denen die von Trident benötigten Container-Images in einer privaten Registry gespeichert sind. Wenn Sie keine private Image-Registry besitzen, verwenden Sie die **"Prozess für die Standardbereitstellung"**.

Wichtige Informationen zu Trident 26.02

Sie müssen die folgenden wichtigen Informationen über Trident lesen.

```

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

```

Wichtige Informationen zu Trident

- Kubernetes 1.35 wird nun in Trident unterstützt. Trident sollte vor Kubernetes aktualisiert werden.
- Trident setzt die Verwendung der Multipathing-Konfiguration in SAN-Umgebungen strikt durch, mit einem empfohlenen Wert von `find_multipaths: no` in der `multipath.conf`-Datei.

Die Verwendung einer Konfiguration ohne Multipathing oder die Verwendung von `find_multipaths: yes` oder `find_multipaths: smart` Wert in der Datei `multipath.conf` führt zu Mount-Fehlern. Trident hat die Verwendung von `find_multipaths: no` seit der Version 21.07 empfohlen.

Den Trident-Operator bereitstellen und Trident mit Helm installieren

Mit dem Trident "[Helm Chart](#)" können Sie den Trident Operator bereitstellen und Trident in einem Schritt installieren.

Überprüfen Sie "[die Installationsübersicht](#)", um sicherzustellen, dass Sie die Installationsvoraussetzungen erfüllt und die richtige Installationsoption für Ihre Umgebung ausgewählt haben.

Bevor Sie beginnen

Zusätzlich zu dem "[Bereitstellungsvoraussetzungen](#)" benötigen Sie "[Helm Version 3](#)".



Wenn Sie Trident in einem privaten Repository installieren und den `imageRegistry` Schalter verwenden, um den Speicherort des Repositories anzugeben, verwenden Sie `/netapp/` nicht im Repository-Pfad.

Schritte

1. Fügen Sie das Trident Helm repository hinzu:

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

2. Verwenden Sie `helm install` und geben Sie einen Namen für Ihre Bereitstellung sowie den Speicherort der Image-Registry an. Ihre "[Trident und CSI images](#)" können sich in einer Registry oder in verschiedenen Registries befinden, aber alle CSI-Images müssen sich in derselben Registry befinden. In den Beispielen ist `100.2602.0` die Version von Trident, die Sie installieren.

Images in einem Registry

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

Images in verschiedenen Registern

```
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}
```



Wenn Sie bereits einen Namespace für Trident erstellt haben, wird durch den `--create-namespace` Parameter kein zusätzlicher Namespace erstellt.

Sie können `helm list` verwenden, um Installationsdetails wie Name, Namespace, Chart, Status, App-Version und Revisionsnummer zu überprüfen.

Konfigurationsdaten während der Installation übergeben

Es gibt zwei Möglichkeiten, Konfigurationsdaten während der Installation zu übergeben:

Option	Beschreibung
<code>--values</code> (oder <code>-f</code>)	Geben Sie eine YAML-Datei mit Überschreibungen an. Dies kann mehrfach angegeben werden, und die ganz rechts stehende Datei hat Vorrang.
<code>--set</code>	Geben Sie Überschreibungen in der Befehlszeile an.

Um beispielsweise den Standardwert von `debug` zu ändern, führen Sie den folgenden Befehl aus, wobei `100.2602.0` die Version von Trident ist, die Sie installieren:

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

Um den `nodePrep` Wert hinzuzufügen, führen Sie den folgenden Befehl aus:

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


Konfigurationsoptionen

Diese Tabelle und die `values.yaml` Datei, die Teil des Helm-Charts ist, enthalten die Liste der Schlüssel und ihrer Standardwerte.





Entfernen Sie die Standardaffinität nicht aus der Datei `values.yaml`. Wenn Sie eine benutzerdefinierte Affinität angeben möchten, erweitern Sie die Standardaffinität.


Option	Beschreibung	Standard
<code>nodeSelector</code>	Knotenbezeichnungen für die Pod-Zuweisung	
<code>podAnnotations</code>	Pod-Anmerkungen	
<code>deploymentAnnotations</code>	Bereitstellungsanmerkungen	
<code>tolerations</code>	Toleranzen für die Pod-Zuordnung	

Option	Beschreibung	Standard
affinity	Affinität für die Pod-Zuweisung	<pre data-bbox="1047 157 1489 1144"> affinity: nodeAffinity: requiredDuringSchedulingIgnoredDuringExecution: nodeSelectorTerms: - matchExpressions: - key: kubernetes.io/arch operator: In values: - arm64 - amd64 - key: kubernetes.io/os operator: In values: - linux </pre> <div data-bbox="1047 1165 1489 1564" style="border: 1px solid #ccc; padding: 10px; margin-top: 10px;">  <p>Entfernen Sie die Standardaffinität nicht aus der Datei values.yaml. Wenn Sie eine benutzerdefinierte Affinität angeben möchten, erweitern Sie die Standardaffinität.</p> </div>
tridentControllerPluginNodeSelector	Zusätzliche Knotenselektoren für Pods. Siehe "Controller-Pods und Node-Pods verstehen" für Details.	
tridentControllerPluginTolerations	Überschreibt die Kubernetes-Toleranzen für Pods. Siehe "Controller-Pods und Node-Pods verstehen" für Details.	

Option	Beschreibung	Standard
<code>tridentNodePluginNodeSelector</code>	Zusätzliche Knotenselektoren für Pods. Siehe " Controller-Pods und Node-Pods verstehen " für Details.	
<code>tridentNodePluginTolerations</code>	Überschreibt die Kubernetes-Toleranzen für Pods. Siehe " Controller-Pods und Node-Pods verstehen " für Details.	
<code>imageRegistry</code>	Identifiziert die Registry für die <code>trident-operator</code> , <code>trident</code> und andere Images. Leer lassen, um den Standardwert zu akzeptieren. WICHTIG: Wenn Sie Trident in einem privaten Repository installieren und den <code>imageRegistry</code> -Schalter zur Angabe des Repository-Speicherorts verwenden, verwenden Sie <code>/netapp/</code> nicht im Repository-Pfad.	""
<code>imagePullPolicy</code>	Legt die Image-Pull-Richtlinie für das <code>trident-operator</code> fest.	<code>IfNotPresent</code>
<code>imagePullSecrets</code>	Legt die Image-Pull-Secrets für die <code>trident-operator</code> , <code>trident</code> und andere Images fest.	
<code>kubeletDir</code>	Ermöglicht das Überschreiben des Host-Speicherorts des internen Zustands von kubelet.	<code>"/var/lib/kubelet"</code>
<code>operatorLogLevel</code>	Ermöglicht das Protokollierungslevel des Trident-Operators auf: <code>trace</code> , <code>debug</code> , <code>info</code> , <code>warn</code> , <code>error</code> oder <code>fatal</code> festzulegen.	<code>"info"</code>
<code>operatorDebug</code>	Ermöglicht das Festlegen des Protokollierungsgrads des Trident-Operators auf Debug.	<code>true</code>
<code>operatorImage</code>	Ermöglicht das vollständige Überschreiben des Bildes für <code>trident-operator</code> .	""
<code>operatorImageTag</code>	Ermöglicht das Überschreiben des Tags des <code>trident-operator</code> -Images.	""
<code>tridentIPv6</code>	Ermöglicht die Aktivierung von Trident für den Einsatz in IPv6-Clustern.	<code>false</code>

Option	Beschreibung	Standard
<code>tridentK8sTimeout</code>	<p>Überschreibt die standardmäßige 180-Sekunden-Zeitüberschreitung für die meisten Kubernetes-API-Operationen (falls ungleich Null, in Sekunden).</p> <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;">  <p>Der <code>tridentK8sTimeout</code> Parameter ist nur für Trident-Installationen anwendbar.</p> </div>	180
<code>tridentHttpRequestTimeout</code>	Überschreibt die standardmäßige 90-Sekunden-Zeitüberschreitung für HTTP-Anfragen, wobei 0s eine unbegrenzte Dauer für die Zeitüberschreitung bedeutet. Negative Werte sind nicht zulässig.	"90s"
<code>tridentSilenceAutosupport</code>	Ermöglicht das Deaktivieren der periodischen AutoSupport-Berichterstattung von Trident.	false
<code>tridentAutosupportImageTag</code>	Ermöglicht das Überschreiben des Tags des Images für den Trident AutoSupport-Container.	<version>
<code>tridentAutosupportProxy</code>	Ermöglicht dem Trident AutoSupport Container, über einen HTTP-Proxy nach Hause zu telefonieren.	""
<code>tridentLogFormat</code>	Legt das Trident Protokollierungsformat (<code>text</code> oder <code>json</code>) fest.	"text"
<code>tridentDisableAuditLog</code>	Deaktiviert den Trident Audit-Logger.	true
<code>tridentLogLevel</code>	Ermöglicht das Protokollierungslevel von Trident auf <code>trace</code> , <code>debug</code> , <code>info</code> , <code>warn</code> , <code>error</code> oder <code>fatal</code> festzulegen.	"info"
<code>tridentDebug</code>	Ermöglicht das Festlegen des Protokollierungslevels von Trident auf <code>debug</code> .	false
<code>tridentLogWorkflows</code>	Ermöglicht die Aktivierung bestimmter Trident-Workflows für die Protokollierung von Ablaufverfolgungen oder die Unterdrückung von Protokollen.	""

Option	Beschreibung	Standard
tridentLogLayers	Ermöglicht die Aktivierung bestimmter Trident-Ebenen für die Protokollierung von Ablaufverfolgungen oder die Unterdrückung von Protokollen.	""
tridentImage	Ermöglicht das vollständige Überschreiben des Images für Trident.	""
tridentImageTag	Ermöglicht das Überschreiben des Tags des Images für Trident.	""
tridentProbePort	Ermöglicht das Überschreiben des Standardports, der für Kubernetes Liveness-/Readiness-Probes verwendet wird.	""
windows	Ermöglicht die Installation von Trident auf einem Windows-Worker-Knoten.	false
enableForceDetach	Ermöglicht das Aktivieren der Funktion „Force Detach“. Sie können den erzwungenen Trennvorgang durch Integration mit dem node health check (NHC) Operator automatisieren. Weitere Informationen finden Sie unter "Automatisierung des Failovers von zustandsbehafteten Anwendungen mit Trident" .	false
excludePodSecurityPolicy	Schließt die Sicherheitsrichtlinie für den Operator-Pod von der Erstellung aus.	false
nodePrep	Ermöglicht Trident, die Knoten des Kubernetes-Clusters für die Verwaltung von Volumes mithilfe des angegebenen Datenspeicherprotokolls vorzubereiten. Derzeit ist <code>iscsi</code> der einzige unterstützte Wert. <div style="border-left: 1px solid #ccc; padding-left: 10px; margin-left: 20px;">  Ab OpenShift 4.19 ist die minimale unterstützte Trident-Version für diese Funktion 25.06.1. </div>	

Option	Beschreibung	Standard
resources	<p>Legt Kubernetes-Ressourcenlimits und -anforderungen für die Trident-Controller-, Node- und Operator-Pods fest. Sie können CPU und Arbeitsspeicher für jeden Container und Sidecar konfigurieren, um die Ressourcenzuweisung in Kubernetes zu verwalten.</p> <p>Weitere Informationen zur Konfiguration von Ressourcenanforderungen und -limits finden Sie unter "Ressourcenverwaltung für Pods und Container".</p> <div style="border-left: 1px solid #ccc; padding-left: 10px; margin-left: 20px;">  <ul style="list-style-type: none"> • Ändern Sie NICHT die Namen von Containern oder Feldern. • Ändern Sie die Einrückung NICHT – die YAML-Einrückung ist für das korrekte Parsen entscheidend. </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: linux:</pre>

Trident-Operator-Installation anpassen

Der Trident-Operator ermöglicht es Ihnen, die Trident-Installation mithilfe der Attribute in der TridentOrchestrator spec anzupassen. Wenn Sie die Installation über das hinaus anpassen möchten, was TridentOrchestrator Argumente erlauben, sollten Sie `tridentctl` verwenden, um benutzerdefinierte YAML-Manifeste zu generieren und diese nach Bedarf zu modifizieren.

Controller-Pods und Node-Pods verstehen

Trident läuft als einzelner Controller-Pod und als Node-Pod auf jedem Worker-Knoten im Cluster. Der Node-Pod muss auf jedem Host ausgeführt werden, auf dem Sie möglicherweise ein Trident Volume einbinden möchten.

Kubernetes "Knotenselektoren" und "Tolerations und Taints" werden verwendet, um einen Pod auf einen bestimmten oder bevorzugten Knoten zu beschränken. Mit dem ControllerPlugin und NodePlugin können Sie Einschränkungen und Überschreibungen festlegen.

- Das Controller-Plugin übernimmt die Bereitstellung und Verwaltung von Volumes, wie Snapshots und Größenänderungen.
- Das Node-Plugin übernimmt das Anbinden des Speicher an den Knoten.

Konfigurationsoptionen




`spec.namespace` wird in TridentOrchestrator angegeben, um den Namespace zu kennzeichnen, in dem Trident installiert ist. Dieser Parameter **kann nach der Installation von Trident nicht mehr aktualisiert werden**. Ein entsprechender Versuch führt dazu, dass sich der TridentOrchestrator Status in Failed ändert. Trident ist nicht für die Migration zwischen Namespaces vorgesehen.



Diese Tabelle enthält `TridentOrchestrator` Attribute im Detail.

Parameter	Beschreibung	Standard
<code>namespace</code>	Namespace, in dem Trident installiert werden soll	"default"
<code>debug</code>	Debugging für Trident aktivieren	false
<code>enableForceDetach</code>	<code>ontap-san</code> , <code>ontap-san-economy</code> , <code>ontap-nas</code> , und <code>ontap-nas-economy</code> nur. Funktioniert mit Kubernetes Non-Graceful Node Shutdown (NGNS), um Clusteradministratoren die Möglichkeit zu geben, Workloads mit eingebundenen Volumes sicher auf neue Knoten zu migrieren, falls ein Knoten fehlerhaft wird. Weitere Informationen finden Sie unter "Automatisierung des Failovers von zustandsbehafteten Anwendungen mit Trident" .	false
<code>windows</code>	Das Setzen auf <code>true</code> ermöglicht die Installation auf Windows-Worker-Knoten.	false

```
trident-main:
  requests:
    cpu: 10m
    memory: 60Mi
  limits:
    cpu:
    memory:
  windows:
  trident-main:
    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:
```

Parameter	Beschreibung	Standard
cloudProvider	Auf "Azure" setzen, wenn verwaltete Identitäten oder eine Cloud-Identität auf einem AKS-Cluster verwendet werden. Setzen Sie auf "AWS", wenn eine Cloud-Identität auf einem EKS-Cluster verwendet wird. Setzen Sie auf "GCP", wenn eine Cloud-Identität auf einem GKE-Cluster verwendet wird.	""
cloudIdentity	Legen Sie auf die Workload-Identität („azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxx“) fest, wenn Sie Cloud Identity auf einem AKS-Cluster verwenden. Legen Sie auf die AWS IAM-Rolle („eks.amazonaws.com/role-arn: arn:aws:iam::123456:role/trident-role“) fest, wenn Sie Cloud Identity auf einem EKS-Cluster verwenden. Auf Cloud-Identität („iam.gke.io/gcp-service-account: xxx@mygcpproject.iam.gserviceaccount.com“) setzen, wenn Cloud Identity auf einem GKE-Cluster verwendet wird.	""
IPv6	Trident über IPv6 installieren	false
k8sTimeout	Zeitüberschreitung für Kubernetes-Operationen.  Der k8sTimeout Parameter ist nur für Trident-Installationen anwendbar.	180sec
silenceAutosupport	Senden Sie Autosupport-Pakete nicht automatisch an NetApp	false
autosupportImage	Das Container-Image für Autosupport Telemetry	"netapp/trident-autosupport10"
autosupportProxy	Die Adresse/der Port eines Proxys zum Senden von Autosupport-Telemetrie	"http://proxy.example.com:8888"
uninstall	Eine Kennzeichnung, die zur Deinstallation von Trident verwendet wird	false
logFormat	Trident Protokollierungsformat, das verwendet werden soll [text,json]	"text"
tridentImage	Trident-Image zur Installation	"netapp/trident:26.02"
imageRegistry	Pfad zur internen Registry im Format <registry FQDN>[:port][:/subpath]	"registry.k8s.io"
kubeletDir	Pfad zum kubelet-Verzeichnis auf dem Host	"/var/lib/kubelet"
wipeout	Eine Liste der Ressourcen, die gelöscht werden müssen, um eine vollständige Entfernung von Trident durchzuführen	

Parameter	Beschreibung	Standard
imagePullSecrets	Geheimnisse zum Abrufen von Images aus einer internen Registry	
imagePullPolicy	Legt die Richtlinie zum Abrufen von Images für den Trident Operator fest. Gültige Werte sind: Always um das Image immer abzurufen. IfNotPresent um das Image nur abzurufen, wenn es noch nicht auf dem Knoten vorhanden ist. Never um das Image niemals abzurufen.	IfNotPresent
controllerPluginNodeSelector	Zusätzliche Knotenselektoren für Pods. Folgt dem gleichen Format wie <code>pod.spec.nodeSelector</code> .	Kein Standardwert; optional
controllerPluginTolerations	Überschreibt die Kubernetes-Toleranzen für Pods. Folgt dem gleichen Format wie <code>pod.spec.Tolerations</code> .	Kein Standardwert; optional
nodePluginNodeSelector	Zusätzliche Knotenselektoren für Pods. Folgt dem gleichen Format wie <code>pod.spec.nodeSelector</code> .	Kein Standardwert; optional
nodePluginTolerations	Überschreibt die Kubernetes-Toleranzen für Pods. Folgt dem gleichen Format wie <code>pod.spec.Tolerations</code> .	Kein Standardwert; optional
nodePrep	Ermöglicht Trident, die Knoten des Kubernetes-Clusters für die Verwaltung von Volumes mithilfe des angegebenen Datenspeicherprotokolls vorzubereiten. Derzeit ist <code>iscsi</code> der einzige unterstützte Wert. <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;">  Ab OpenShift 4.19 ist die minimale unterstützte Trident-Version für diese Funktion 25.06.1. </div>	
k8sAPIQPS	Die vom Controller bei der Kommunikation mit dem Kubernetes-API-Server verwendete Abfrageanzahl pro Sekunde (QPS). Der Burst-Wert wird automatisch anhand des QPS-Werts festgelegt.	100; optional
enableConcurrency	Ermöglicht gleichzeitige Trident-Controller-Operationen für einen verbesserten Durchsatz. <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;">  Tech Preview: Diese Funktion ist experimentell und unterstützt derzeit eingeschränkte parallele Arbeitsabläufe mit den ONTAP-NAS (nur NFS) und ONTAP-SAN (NVMe für unified ONTAP 9) Treibern, zusätzlich zur bestehenden Tech Preview für den ONTAP-SAN Treiber (iSCSI- und FCP-Protokolle in unified ONTAP 9). </div>	false

Parameter	Beschreibung	Standard
resources	<p>Legt Ressourcenlimits und -anforderungen für den Trident-Controller und die Node-Pods in Kubernetes fest. Sie können CPU und Arbeitsspeicher für jeden Container und Sidecar konfigurieren, um die Ressourcenzuweisung in Kubernetes zu verwalten.</p> <p>Weitere Informationen zur Konfiguration von Ressourcenanforderungen und -limits finden Sie unter "Ressourcenverwaltung für Pods und Container".</p> <ul style="list-style-type: none"> • Ändern Sie NICHT die Namen von Containern oder Feldern. • Ändern Sie die Einrückung NICHT – die YAML-Einrückung ist für das korrekte Parsen entscheidend. • Standardmäßig werden keine Limits angewendet – nur Anfragen haben Standardwerte und werden automatisch angewendet, wenn sie nicht angegeben sind. • Containernamen werden so aufgelistet, wie sie in den Pod-Spezifikationen erscheinen. • Sidecars sind unter jedem Hauptcontainer aufgeführt. • Überprüfen Sie das Feld „TORC“ <code>status.CurrentInstallation.Params</code>, um die aktuell angewendeten Werte anzuzeigen. 	<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:</pre>

Parameter	Beschreibung	Standard
httpsMetrics	Aktivieren Sie HTTPS für den Prometheus metrics endpoint.	false
hostNetwork	Aktiviert die Host-Netzwerkverbindung für den Trident Controller. Dies ist nützlich, wenn Sie den Frontend- und Backend-Datenverkehr in einem Multi-Home-Netzwerk trennen möchten.	false



Weitere Informationen zur Formatierung von Pod-Parametern finden Sie unter "[Pods, Knoten zuweisen](#)".

```

30Mi
limits:
  cpu:
  memory:
node:
linux:
trident-
main:

```

Beispielkonfigurationen

Sie können die Attribute in [Konfigurationsoptionen](#) verwenden, wenn Sie `TridentOrchestrator` definieren, um Ihre Installation anzupassen.

Grundlegende benutzerdefinierte Konfiguration

Dieses Beispiel, das nach Ausführung des `cat deploy/crds/tridentorchestrator_cr_imagepullsecrets.yaml` Befehls erstellt wurde, stellt eine einfache benutzerdefinierte Installation dar:

```

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

```

```

lm
memory: 10Mi
  limits:
    cpu:
memory:
  windows:
    trident-
main:
requests:
  cpu:
6m
memory: 40Mi
  limits:

```

Knotenselektoren

Dieses Beispiel installiert Trident mit Knotenselektoren.

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

```
memory:
liveness-
```

Windows-Worker-Knoten

Dieses Beispiel, das nach Ausführung des `cat deploy/crds/tridentorchestrator_cr.yaml` Befehls erstellt wurde, installiert Trident auf einem Windows-Worker-Knoten.

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

Verwaltete Identitäten auf einem AKS-Cluster

Dieses Beispiel installiert Trident, um verwaltete Identitäten auf einem AKS-Cluster zu aktivieren.

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

Cloud-Identität auf einem AKS-Cluster

Dieses Beispiel installiert Trident zur Verwendung mit einer Cloud-Identität auf einem AKS-Cluster.

```
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-xxxxxxxxxxxxx'
```

Cloud-Identität auf einem EKS-Cluster

Dieses Beispiel installiert Trident zur Verwendung mit einer Cloud-Identität auf einem AKS-Cluster.

```
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'"
```

Cloud-Identität für GKE

Dieses Beispiel installiert Trident zur Verwendung mit einer Cloud-Identität auf einem GKE-Cluster.

```
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
```

Konfiguration der Kubernetes-Ressourcenanforderungen und -limits für Trident controller und Trident Linux node pods

Dieses Beispiel konfiguriert Kubernetes-Ressourcenanforderungen und -limits für Trident controller und Trident Linux node pods.



Hinweis: Die in diesem Beispiel angegebenen Anforderungs- und Grenzwerte dienen lediglich Demonstrationszwecken. Passen Sie diese Werte an Ihre Umgebung und Arbeitslastanforderungen an.

```
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
```

Konfiguration von Kubernetes-Ressourcenanforderungen und -limits für Trident Controller sowie für Trident Windows- und Linux-Node-Pods

Dieses Beispiel konfiguriert Kubernetes-Ressourcenanforderungen und -Limits für Trident Controller sowie Trident Windows- und Linux Node Pods.



Hinweis: Die in diesem Beispiel angegebenen Anforderungs- und Grenzwerte dienen lediglich Demonstrationszwecken. Passen Sie diese Werte an Ihre Umgebung und Arbeitslastanforderungen an.

```
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
```

Installation mit tridentctl

Installation mit tridentctl

Sie können Trident mit `tridentctl` installieren. Dieser Prozess gilt für Installationen, bei denen die von Trident benötigten Container-Images entweder in einer privaten Registry gespeichert sind oder nicht. Um Ihre `tridentctl` Bereitstellung anzupassen, siehe "[tridentctl-Bereitstellung anpassen](#)".

Wichtige Informationen zu Trident10

Sie müssen die folgenden wichtigen Informationen über Trident lesen.

Wichtige Informationen zu Trident

- Kubernetes 1.27 wird jetzt in Trident unterstützt. Trident sollte vor Kubernetes aktualisiert werden.
- Trident setzt die Verwendung der Multipathing-Konfiguration in SAN-Umgebungen strikt durch, mit einem empfohlenen Wert von `find_multipaths: no` in der `multipath.conf`-Datei.

Die Verwendung einer Konfiguration ohne Multipathing oder die Verwendung von `find_multipaths: yes` oder `find_multipaths: smart` Wert in der Datei `multipath.conf` führt zu Mount-Fehlern. Trident hat die Verwendung von `find_multipaths: no` seit der Version 21.07 empfohlen.

Installieren Sie Trident mit `tridentctl`

Überprüfen Sie "[die Installationsübersicht](#)", um sicherzustellen, dass Sie die Installationsvoraussetzungen erfüllt und die richtige Installationsoption für Ihre Umgebung ausgewählt haben.

Bevor Sie beginnen

Bevor Sie mit der Installation beginnen, melden Sie sich am Linux-Host an und vergewissern Sie sich, dass dieser ein funktionierendes "[unterstützter Kubernetes-Cluster](#)" verwaltet und dass Sie über die erforderlichen

Berechtigungen verfügen.



Mit OpenShift verwenden Sie `oc` statt `kubectl` in allen folgenden Beispielen und melden Sie sich zuerst als **system:admin** an, indem Sie `oc login -u system:admin` oder `oc login -u kube-admin` ausführen.

1. Überprüfen Sie Ihre Kubernetes-Version:

```
kubectl version
```

2. Überprüfen Sie die Cluster-Administratorrechte:

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

3. Überprüfen Sie, ob Sie einen Pod starten können, der ein Image von Docker Hub verwendet, und Ihr Speichersystem über das Pod-Netzwerk erreichen können.

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

Schritt 1: Laden Sie das Trident-Installationspaket herunter

Das Trident-Installationspaket erstellt einen Trident Pod, konfiguriert die CRD-Objekte, die zur Zustandsverwaltung verwendet werden, und initialisiert die CSI-Sidecars, um Aktionen wie das Bereitstellen und Anbinden von Volumes an die Cluster-Hosts auszuführen. Laden Sie die neueste Version des Trident-Installationsprogramms von "[der Abschnitt Assets auf GitHub](#)" herunter und extrahieren Sie sie. Aktualisieren Sie `<trident-installer-XX.XX.X.tar.gz>` im Beispiel mit Ihrer ausgewählten Trident Version.

```
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
```

Schritt 2: Trident installieren

Installieren Sie Trident im gewünschten Namespace, indem Sie den `tridentctl install` Befehl ausführen. Sie können zusätzliche Argumente hinzufügen, um den Speicherort der Image-Registry anzugeben.

Standardmodus

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

Images in einem Registry

```
./tridentctl install -n trident --image-registry <your-registry>  
--autosupport-image <your-registry>/trident-autosupport:26.02 --trident  
-image <your-registry>/trident:26.02.0
```

Images in verschiedenen Registern

```
./tridentctl install -n trident --image-registry <your-registry>  
--autosupport-image <your-registry>/trident-autosupport:26.02 --trident  
-image <your-registry>/trident:26.02.0
```

Ihr Installationsstatus sollte in etwa so aussehen.

```
....  
INFO Starting Trident installation.                namespace=trident  
INFO Created service account.  
INFO Created cluster role.  
INFO Created cluster role binding.  
INFO Added finalizers to custom resource definitions.  
INFO Created Trident service.  
INFO Created Trident secret.  
INFO Created Trident deployment.  
INFO Created Trident daemonset.  
INFO Waiting for Trident pod to start.  
INFO Trident pod started.                          namespace=trident  
pod=trident-controller-679648bd45-cv2mx  
INFO Waiting for Trident REST interface.  
INFO Trident REST interface is up.                version=26.10.0  
INFO Trident installation succeeded.  
....
```

Überprüfen Sie die Installation

Sie können Ihre Installation anhand des pod creation status oder `tridentctl` überprüfen.

Verwendung des Pod-Erstellungsstatus

Sie können bestätigen, ob die Trident Installation abgeschlossen wurde, indem Sie den Status der erstellten Pods überprüfen:

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

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



Falls das Installationsprogramm nicht erfolgreich abgeschlossen wird oder `trident-controller-<generated id>` (`trident-csi-<generated id>` in Versionen vor 23.01) keinen **Running**-Status hat, wurde die Plattform nicht installiert. Verwenden Sie `-d`, um "[Debug-Modus aktivieren](#)" das Problem zu beheben und zu analysieren.

Verwendung `tridentctl`

Sie können `tridentctl` verwenden, um die installierte Version von Trident zu überprüfen.

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

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

Beispielkonfigurationen

Die folgenden Beispiele enthalten Beispielkonfigurationen für die Installation von Trident mit `tridentctl`.

Windows-Knoten

Um Trident auf Windows-Knoten auszuführen:

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

Trennung erzwingen

Weitere Informationen finden Sie unter "[Automatisierung des Failovers von zustandsbehafteten Anwendungen mit Trident](#)".

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

Gleichzeitige Trident-Controller-Operationen ermöglichen

Um gleichzeitige Trident-Controller-Operationen für einen verbesserten Durchsatz zu ermöglichen, fügen Sie die `--enable-concurrency` Option während der Installation wie in diesem Beispiel gezeigt hinzu.



Tech Preview: Diese Funktion ist experimentell und unterstützt derzeit eingeschränkte parallele Arbeitsabläufe mit den ONTAP-NAS (nur NFS) und ONTAP-SAN (NVMe für unified ONTAP 9) Treibern, zusätzlich zur bestehenden Tech Preview für den ONTAP-SAN Treiber (iSCSI- und FCP-Protokolle in unified ONTAP 9).

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

tridentctl-Installation anpassen

Sie können das Trident-Installationsprogramm verwenden, um die Installation anzupassen.

Erfahren Sie mehr über das Installationsprogramm

Der Trident-Installer ermöglicht es Ihnen, Attribute anzupassen. Wenn Sie beispielsweise das Trident-Image in ein privates Repository kopiert haben, können Sie den Image-Namen mit `--trident-image` angeben. Wenn Sie sowohl das Trident-Image als auch die benötigten CSI-Sidecar-Images in ein privates Repository kopiert haben, kann es vorzuziehen sein, den Speicherort dieses Repositories mit dem `--image-registry`-Schalter anzugeben, der die Form `<registry FQDN>[:port]` hat.



Wenn Sie Trident in einem privaten Repository installieren und den `--image-registry` Schalter verwenden, um den Speicherort des Repositories anzugeben, verwenden Sie `/netapp/` nicht im Repository-Pfad. Zum Beispiel: `./tridentctl install --image-registry <image-registry> -n <namespace>`

Wenn Sie eine Distribution von Kubernetes verwenden, bei der `kubelet` seine Daten auf einem anderen Pfad als dem üblichen `/var/lib/kubelet` speichert, können Sie den alternativen Pfad mit `--kubelet-dir` angeben.

Falls Sie die Installation über die Argumente des Installationsprogramms hinaus anpassen müssen, können Sie auch die Bereitstellungsdateien anpassen. Die Verwendung des `--generate-custom-yaml` Parameters erstellt die folgenden YAML-Dateien im `setup` Verzeichnis des Installationsprogramms:

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

- trident-serviceaccount.yaml
- trident-resourcequota.yaml *

Nachdem Sie diese Dateien generiert haben, können Sie sie entsprechend Ihren Bedürfnissen anpassen und dann `--use-custom-yaml` verwenden, um Ihre benutzerdefinierte Bereitstellung zu installieren.

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

Installation mit dem OpenShift zertifizierten Operator

Installieren Sie Trident mit OpenShift OperatorHub

Wenn Sie Red Hat OpenShift verwenden, können Sie NetApp Trident mithilfe des von Red Hat zertifizierten Operators installieren. Verwenden Sie dieses Verfahren, um Trident von der Red Hat OpenShift Container Platform zu installieren.

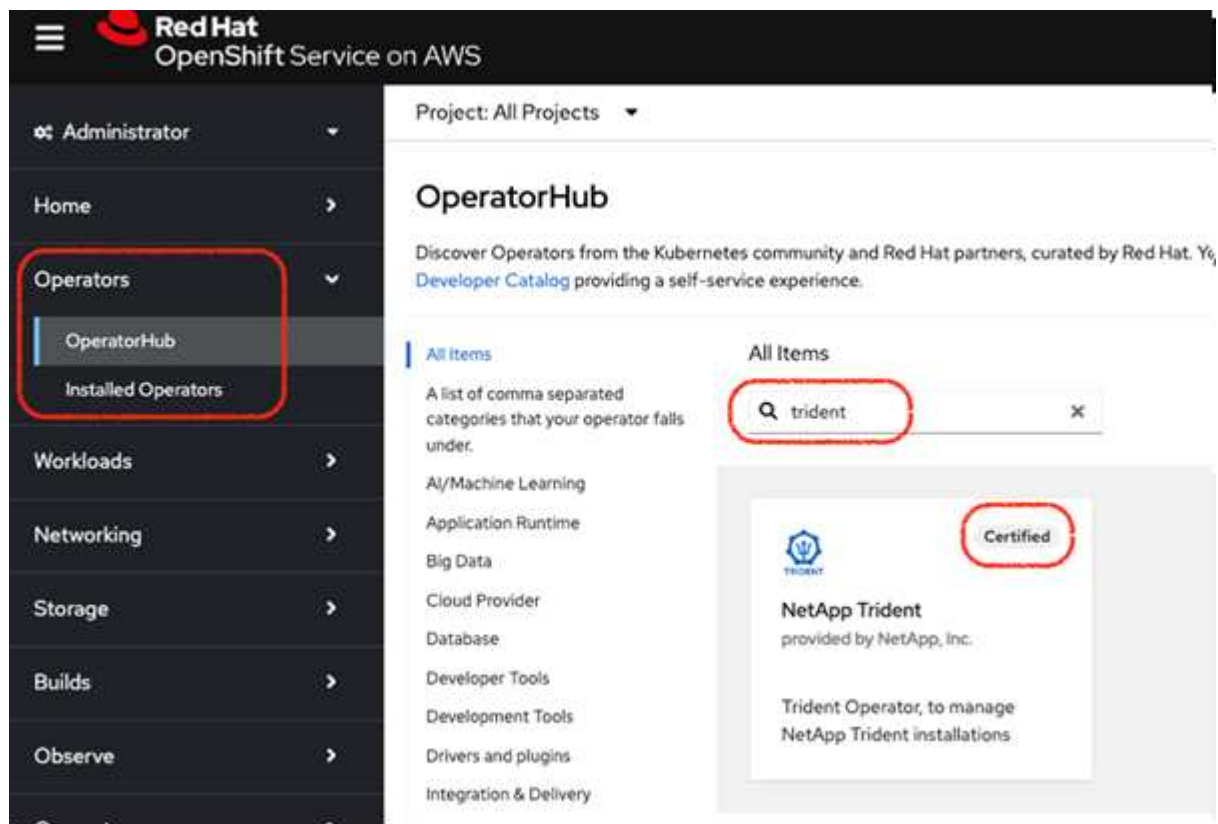
Bevor Sie beginnen

Bevor Sie mit der Installation beginnen, "[Bereiten Sie Ihre Umgebung für die Trident Installation vor](#)".

Suchen und installieren Sie den Trident Operator

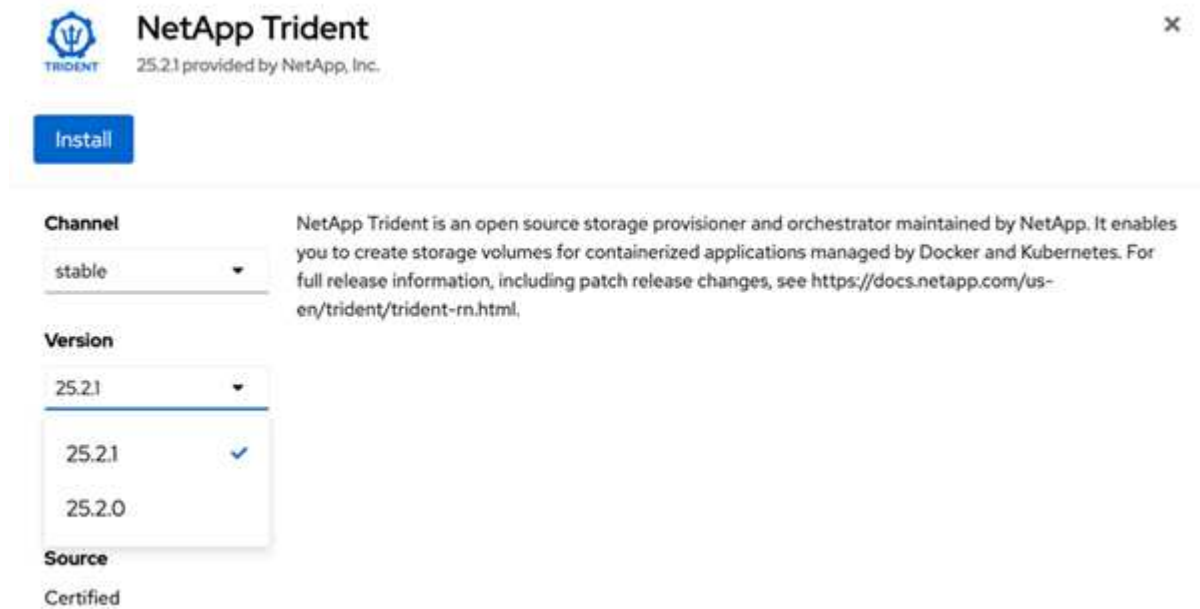
Schritte

1. Navigieren Sie zu OpenShift OperatorHub und suchen Sie nach NetApp Trident.



2. Klicken Sie auf **NetApp Trident**, um die Installationseinstellungen zu öffnen.

3. Wählen Sie die gewünschten Optionen aus und klicken Sie auf **Install**, um die Operator-Konfiguration zu öffnen.



NetApp Trident
25.2.1 provided by NetApp, Inc.

Install

Channel
stable

Version
25.2.1

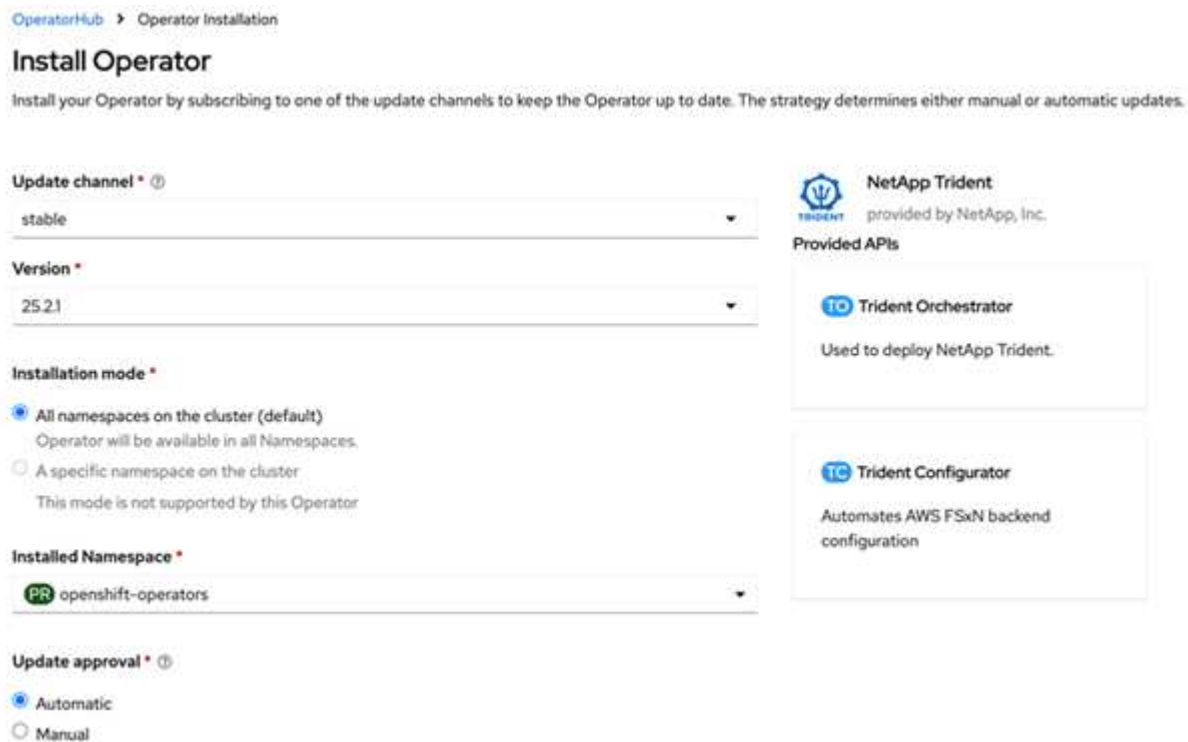
NetApp Trident is an open source storage provisioner and orchestrator maintained by NetApp. It enables you to create storage volumes for containerized applications managed by Docker and Kubernetes. For full release information, including patch release changes, see <https://docs.netapp.com/us-en/trident/trident-rn.html>.

Source
Certified



Stellen Sie sicher, dass Sie die aktuellste Operator-Version auswählen.

4. Behalten Sie alle Parameter bei und klicken Sie auf **Install**.



OperatorHub > Operator Installation

Install Operator

Install your Operator by subscribing to one of the update channels to keep the Operator up to date. The strategy determines either manual or automatic updates.

Update channel ⓘ
stable

Version *
25.2.1

Installation mode *
 All namespaces on the cluster (default)
Operator will be available in all Namespaces.
 A specific namespace on the cluster
This mode is not supported by this Operator

Installed Namespace *
PR openshift-operators

Update approval ⓘ
 Automatic
 Manual

NetApp Trident
provided by NetApp, Inc.

Provided APIs

- TO Trident Orchestrator**
Used to deploy NetApp Trident.
- TC Trident Configurator**
Automates AWS FSxN backend configuration

5. Klicken Sie auf **View Operator**, um die Details des Operators anzuzeigen.



Provided APIs

TO Trident Orchestrator

Used to deploy NetApp Trident.

[Create instance](#)

TC Trident Configurator

Automates AWS FSxN backend configuration

[Create instance](#)

6. Klicken Sie auf **YAML view** und fügen Sie Folgendes in das Formular ein:

```

apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
  namespace: openshift-operators
spec:
  IPv6: false
  debug: false
  nodePrep:
  - iscsi
  imageRegistry: ''
  k8sTimeout: 180s
  namespace: trident
  silenceAutosupport: false

```



Die Benutzeroberfläche bietet ein Standardbeispiel. Sie können dieses direkt bearbeiten, anstatt eine vollständige Konfiguration zu kopieren.

Optional: Gleichzeitigkeit aktivieren



Um Parallelverarbeitung zu ermöglichen, fügen Sie der Spezifikation das folgende Feld hinzu:

```
enableConcurrency: true
```



- Red Hat Enterprise Linux CoreOS (RHCOS) hat iSCSI nicht aktiviert und konfiguriert.
- Sie können den `nodePrep` Parameter hinzufügen, um sowohl iSCSI- als auch Multipath-Dienste auf allen OpenShift Worker-Knoten zu konfigurieren und zu aktivieren.
- Ab OpenShift 4.19 ist die minimale unterstützte Trident-Version für diese Funktion 25.06.1.

1. Klicken Sie auf **Create**; der Trident Orchestrator wird vollständig installiert.

Installed Operators > Operator details

NetApp Trident
25.2.1 provided by NetApp, Inc.

Actions

Details | YAML | Subscription | Events | All instances | **Trident Orchestrator** | Trident Configurator

TridentOrchestrators Create TridentOrchestrator

Name Search by name...

Name	Kind	Status	Labels	Last updated
trident	TridentOrchestrator	Status: Installed	No labels	Apr 6, 2025, 8:02 PM

Trident Operator deinstallieren

Schritte

1. Wählen Sie den Trident Operator aus der Liste der installierten Operator aus.
2. Wählen Sie aus, wenn Sie alle Operandeninstanzen aus dem Operator löschen möchten.



Wenn Sie das Kontrollkästchen **Alle Operandeninstanzen dieses Operators löschen** nicht auswählen, wird Trident nicht deinstalliert.

3. Klicken Sie auf **Uninstall**.

Wechseln Sie zum OpenShift zertifizierten Trident Operator

Sie können vom Community-Operator, einer Helm-basierten Installation oder einem manuell bereitgestellten Operator zum Red Hat OpenShift zertifizierten Trident-Operator wechseln. Der Vorgang umfasst bei jeder Methode die Deinstallation des vorhandenen Operators und anschließend die Installation des zertifizierten Operators mithilfe des OperatorHub.

Bevor Sie beginnen

Bevor Sie mit der Installation beginnen, ["Bereiten Sie Ihre Umgebung für die Trident Installation vor"](#).

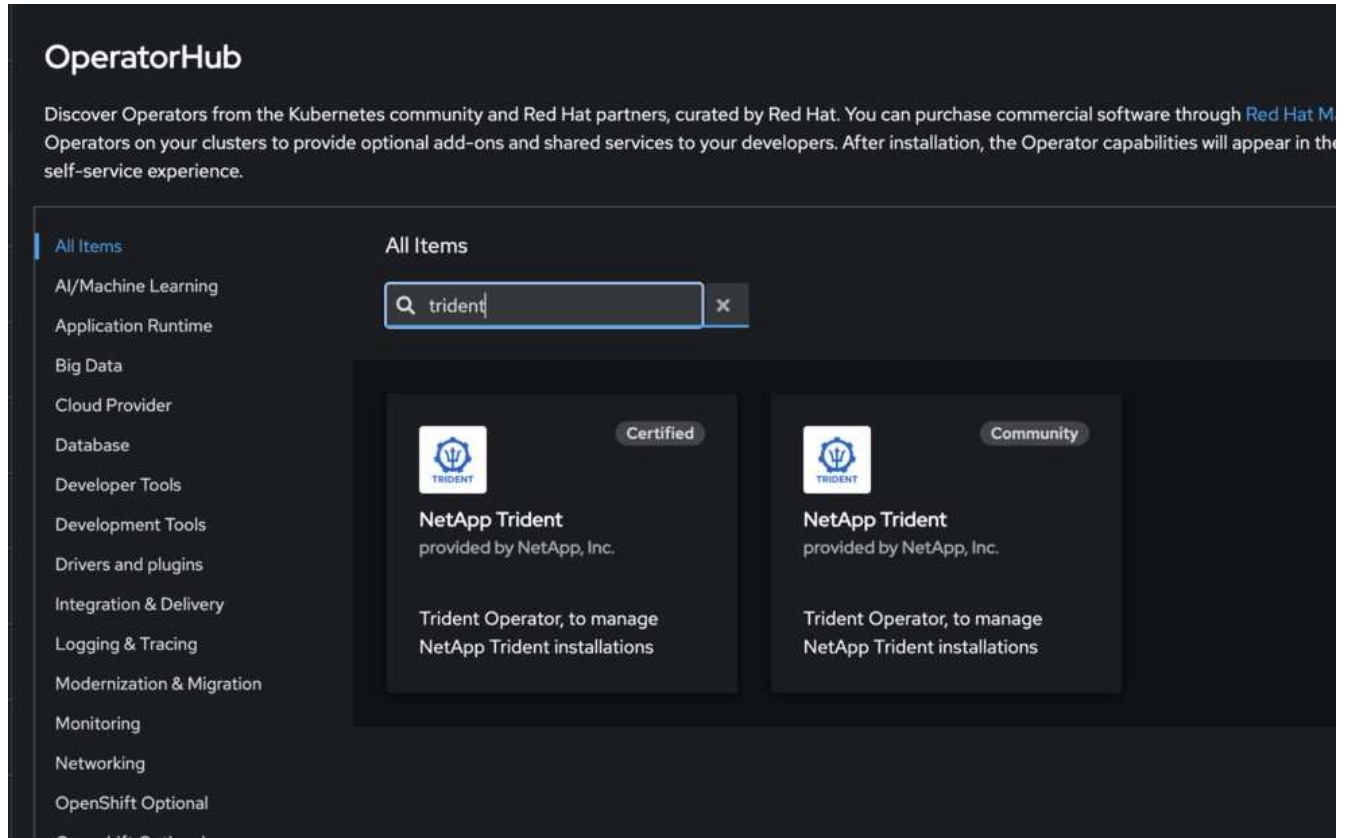


Löschen Sie die `TridentOrchestrator` benutzerdefinierte Ressource (CR) während des Deinstallationsvorgangs nicht. Die `TridentOrchestrator` CR bewahrt Ihre Backend- und Speicherklassenkonfiguration, die nach der Installation des zertifizierten Operators erforderlich ist.

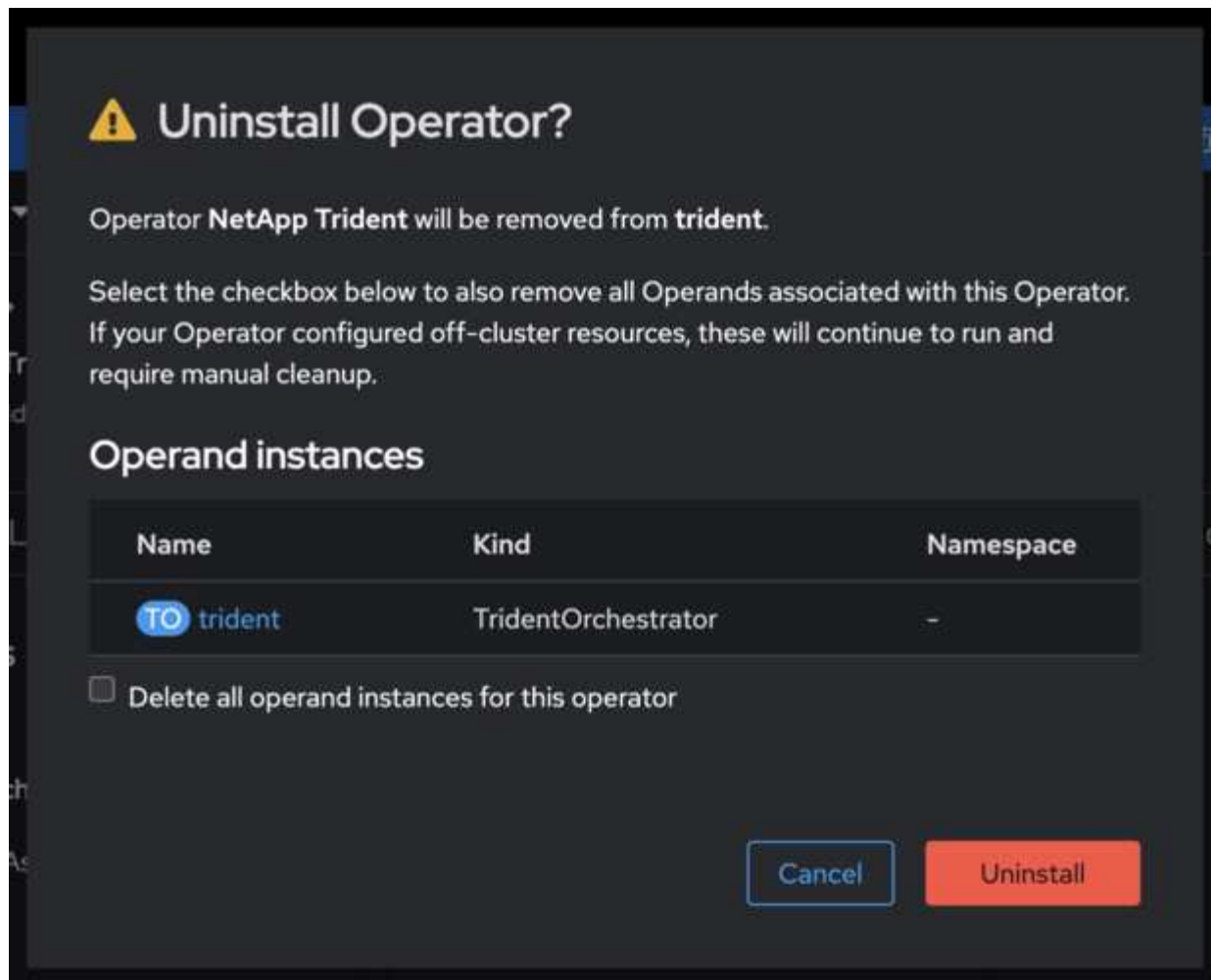
Wechseln Sie vom Community Operator

Schritte

1. Verwenden Sie die OpenShift-Konsole, um zum OperatorHub zu navigieren.



2. Finden Sie den NetApp Trident Community Operator.



Wählen Sie nicht **Alle Operandeninstanzen dieses Operators löschen** aus.

3. Klicken Sie auf **Uninstall**.
4. Nach Abschluss der Deinstallation fahren Sie mit [Installieren Sie den OpenShift zertifizierten Operator](#) fort.

Wechsel von einer Helm-basierten Operator-Installation

Schritte

1. Listen Sie das Helm-Release für Ihre Trident Installation auf:

```
helm ls -n trident
```

2. Deinstallieren Sie das Helm-Release:

```
helm uninstall <release-name> -n trident
```

3. Nach Abschluss der Deinstallation fahren Sie mit [Installieren Sie den OpenShift zertifizierten Operator](#) fort.

Wechsel von einem manuell bereitgestellten Operator

Wenn Sie Trident durch manuelle Bereitstellung des Operators mithilfe eines `bundle.yaml` aus dem Installationspaket installiert haben, entfernen Sie ihn, indem Sie dasselbe Manifest löschen.

Schritte

1. Löschen Sie die Operator-Bereitstellung mithilfe des Bundle-Manifests:

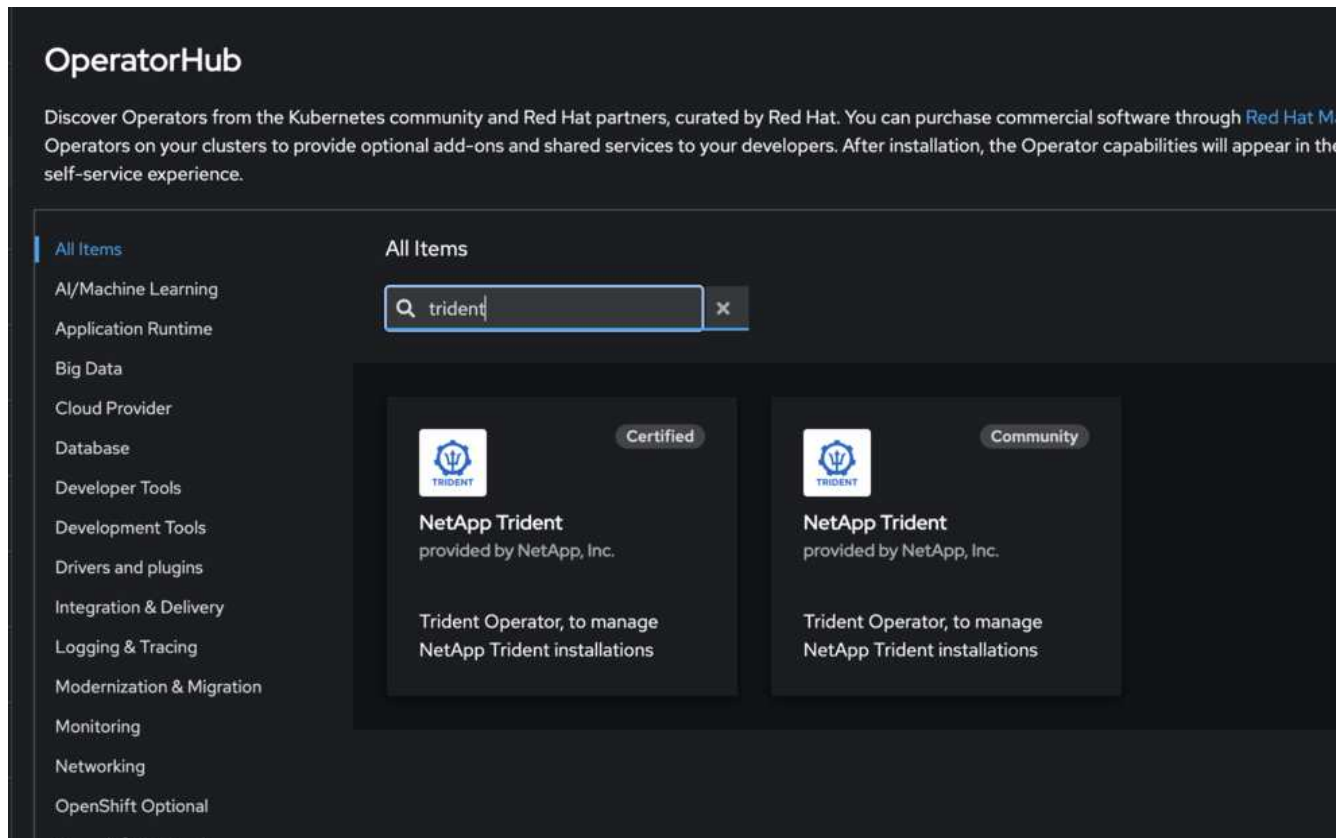
```
kubectl delete -f deploy/bundle.yaml -n trident
```

2. Nach Abschluss der Deinstallation fahren Sie mit [Installieren Sie den OpenShift zertifizierten Operator](#) fort.

Installieren Sie den OpenShift zertifizierten Operator

Schritte

1. Navigieren Sie zum Red Hat OperatorHub.
2. Suchen Sie nach dem NetApp Trident Operator und wählen Sie ihn aus.



3. Folgen Sie den Anweisungen auf dem Bildschirm, um den Operator zu installieren.

Überprüfung

- Überprüfen Sie den OperatorHub in der Konsole, um sicherzustellen, dass der neue zertifizierte Operator erfolgreich installiert wurde.

Copyright-Informationen

Copyright © 2026 NetApp. Alle Rechte vorbehalten. Gedruckt in den USA. Dieses urheberrechtlich geschützte Dokument darf ohne die vorherige schriftliche Genehmigung des Urheberrechtlich geschützten Urhebers in keiner Form und durch keine Mittel – weder grafische noch elektronische oder mechanische, einschließlich Fotokopieren, Aufnehmen oder Speichern in einem elektronischen Abrufsystem – auch nicht in Teilen, vervielfältigt werden.

Software, die von urheberrechtlich geschütztem NetApp Material abgeleitet wird, unterliegt der folgenden Lizenz und dem folgenden Haftungsausschluss:

DIE VORLIEGENDE SOFTWARE WIRD IN DER VORLIEGENDEN FORM VON NETAPP ZUR VERFÜGUNG GESTELLT, D. H. OHNE JEGLICHE EXPLIZITE ODER IMPLIZITE GEWÄHRLEISTUNG, EINSCHLIESSLICH, JEDOCH NICHT BESCHRÄNKT AUF DIE STILLSCHWEIGENDE GEWÄHRLEISTUNG DER MARKTGÄNGIGKEIT UND EIGNUNG FÜR EINEN BESTIMMTEN ZWECK, DIE HIERMIT AUSGESCHLOSSEN WERDEN. NETAPP ÜBERNIMMT KEINERLEI HAFTUNG FÜR DIREKTE, INDIREKTE, ZUFÄLLIGE, BESONDERE, BEISPIELHAFT SCHÄDEN ODER FOLGESCHÄDEN (EINSCHLIESSLICH, JEDOCH NICHT BESCHRÄNKT AUF DIE BESCHAFFUNG VON ERSATZWAREN ODER -DIENSTLEISTUNGEN, NUTZUNGS-, DATEN- ODER GEWINNVERLUSTE ODER UNTERBRECHUNG DES GESCHÄFTSBETRIEBS), UNABHÄNGIG DAVON, WIE SIE VERURSACHT WURDEN UND AUF WELCHER HAFTUNGSTHEORIE SIE BERUHEN, OB AUS VERTRAGLICH FESTGELEGTER HAFTUNG, VERSCHULDENSUNABHÄNGIGER HAFTUNG ODER DELIKTSHAFTUNG (EINSCHLIESSLICH FAHRLÄSSIGKEIT ODER AUF ANDEREM WEGE), DIE IN IRGEND EINER WEISE AUS DER NUTZUNG DIESER SOFTWARE RESULTIEREN, SELBST WENN AUF DIE MÖGLICHKEIT DERARTIGER SCHÄDEN HINGEWIESEN WURDE.

NetApp behält sich das Recht vor, die hierin beschriebenen Produkte jederzeit und ohne Vorankündigung zu ändern. NetApp übernimmt keine Verantwortung oder Haftung, die sich aus der Verwendung der hier beschriebenen Produkte ergibt, es sei denn, NetApp hat dem ausdrücklich in schriftlicher Form zugestimmt. Die Verwendung oder der Erwerb dieses Produkts stellt keine Lizenzierung im Rahmen eines Patentrechts, Markenrechts oder eines anderen Rechts an geistigem Eigentum von NetApp dar.

Das in diesem Dokument beschriebene Produkt kann durch ein oder mehrere US-amerikanische Patente, ausländische Patente oder anhängige Patentanmeldungen geschützt sein.

ERLÄUTERUNG ZU „RESTRICTED RIGHTS“: Nutzung, Vervielfältigung oder Offenlegung durch die US-Regierung unterliegt den Einschränkungen gemäß Unterabschnitt (b)(3) der Klausel „Rights in Technical Data – Noncommercial Items“ in DFARS 252.227-7013 (Februar 2014) und FAR 52.227-19 (Dezember 2007).

Die hierin enthaltenen Daten beziehen sich auf ein kommerzielles Produkt und/oder einen kommerziellen Service (wie in FAR 2.101 definiert) und sind Eigentum von NetApp, Inc. Alle technischen Daten und die Computersoftware von NetApp, die unter diesem Vertrag bereitgestellt werden, sind gewerblicher Natur und wurden ausschließlich unter Verwendung privater Mittel entwickelt. Die US-Regierung besitzt eine nicht ausschließliche, nicht übertragbare, nicht unterlizenzierbare, weltweite, limitierte unwiderrufliche Lizenz zur Nutzung der Daten nur in Verbindung mit und zur Unterstützung des Vertrags der US-Regierung, unter dem die Daten bereitgestellt wurden. Sofern in den vorliegenden Bedingungen nicht anders angegeben, dürfen die Daten ohne vorherige schriftliche Genehmigung von NetApp, Inc. nicht verwendet, offengelegt, vervielfältigt, geändert, aufgeführt oder angezeigt werden. Die Lizenzrechte der US-Regierung für das US-Verteidigungsministerium sind auf die in DFARS-Klausel 252.227-7015(b) (Februar 2014) genannten Rechte beschränkt.

Markeninformationen

NETAPP, das NETAPP Logo und die unter <http://www.netapp.com/TM> aufgeführten Marken sind Marken von NetApp, Inc. Andere Firmen und Produktnamen können Marken der jeweiligen Eigentümer sein.