



# Installation über den Trident Operator

Astra Trident

NetApp  
April 03, 2024

# Inhalt

- Installation über den Trident Operator ..... 1
  - Manuelle Implementierung des Trident-Mitarbeiters (Standard-Modus) ..... 1
  - Manuelles Bereitstellen des Trident-Mitarbeiters (Offline-Modus) ..... 6
  - Trident Operator mit Helm (Standard-Modus) implementieren ..... 12
  - Trident-Operator mit Helm (Offline-Modus) implementieren ..... 17
  - Anpassen der Trident Operator-Installation ..... 22

# Installation über den Trident Operator

## Manuelle Implementierung des Trident-Mitarbeiters (Standard-Modus)

Sie können den Trident-Operator manuell implementieren, um Astra Trident zu installieren. Dieser Prozess gilt für Installationen, bei denen die von Astra Trident benötigten Container-Images nicht in einer privaten Registrierung gespeichert werden. Wenn Sie über eine private Bildregistrierung verfügen, verwenden Sie das ["Prozess für Offline-Implementierung"](#).

### Kritische Informationen zu Astra Trident 23.07

Sie müssen die folgenden wichtigen Informationen über Astra Trident lesen.

#### **Informationen über Astra TriperelT**

- Kubernetes 1.27 wird jetzt in Trident unterstützt. Upgrade von Trident vor dem Upgrade von Kubernetes.
- Astra Trident setzt die Verwendung von Multipathing-Konfiguration in SAN-Umgebungen strikt um und empfiehlt den Nutzen von `find_multipaths: no` In Multipath.conf Datei.

Verwendung einer Konfiguration ohne Multipathing oder Verwendung von `find_multipaths: yes` Oder `find_multipaths: smart` Der Wert in der Multipath.conf-Datei führt zu Mount-Fehlern. Trident empfiehlt die Verwendung von `find_multipaths: no` Seit der Version 21.07.

### Trident-Operator kann manuell implementiert und Trident installiert werden

Prüfen ["Die Übersicht über die Installation"](#) Um sicherzustellen, dass Sie die Installationsvoraussetzungen erfüllt haben, und die richtige Installationsoption für Ihre Umgebung ausgewählt haben.

#### Bevor Sie beginnen

Melden Sie sich vor der Installation beim Linux-Host an, und überprüfen Sie, ob er einen funktionierenden ["Unterstützter Kubernetes-Cluster"](#) Und dass Sie die erforderlichen Berechtigungen haben.



Mit OpenShift, verwenden `oc` Statt `kubectl` In allen folgenden Beispielen, und melden Sie sich als **System:admin** zuerst mit dem Ausführen `oc login -u system:admin` Oder `oc login -u kube-admin`.

### 1. Überprüfen Sie Ihre Kubernetes Version:

```
kubectl version
```

### 2. Überprüfung der Berechtigungen für Cluster-Administratoren:

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

### 3. Überprüfen Sie, ob Sie einen Pod starten können, der ein Image aus dem Docker Hub verwendet, und ob er das Storage-System über das POD-Netzwerk erreichen kann:

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

## Schritt 1: Laden Sie das Trident Installer-Paket herunter

Das Astra Trident Installationspaket enthält alles, was Sie für die Bereitstellung des Trident-Operators und die Installation von Astra Trident benötigen. Laden Sie die neueste Version des Trident Installationsprogramms herunter und extrahieren Sie sie aus ["Die Sektion Assets auf GitHub"](#).

```
wget https://github.com/NetApp/trident/releases/download/v23.07.1/trident-
installer-23.07.1.tar.gz
tar -xf trident-installer-23.07.1.tar.gz
cd trident-installer
```

## Schritt 2: Erstellen Sie die TridentOrchestrator CRD.

Erstellen Sie die TridentOrchestrator Benutzerdefinierte Ressourcendefinition (CRD). Sie werden ein erstellen TridentOrchestrator Benutzerdefinierte Ressourcen später. Verwenden Sie die entsprechende CRD YAML-Version in `deploy/crds` Um die zu erstellen TridentOrchestrator CRD.-

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

## Schritt 3: Implementieren Sie den Trident-Operator

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

- Verwenden Sie für Cluster mit Kubernetes 1.24 oder früheren Versionen `bundle_pre_1_25.yaml`.

- Verwenden Sie für Cluster mit Kubernetes 1.25 oder höher `bundle_post_1_25.yaml`.

### Bevor Sie beginnen

- Standardmäßig stellt das Trident-Installationsprogramm den Operator in bereit `trident` Namespace. Wenn der `trident` Namespace ist nicht vorhanden, erstellen Sie ihn mit:

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

- Um den Operator in einem anderen Namespace als dem bereitzustellen `trident` Namespace, Update `serviceaccount.yaml`, `clusterrolebinding.yaml` Und `operator.yaml` Und erstellen Sie Ihre Bundle-Datei mit `kustomization.yaml`.
  - a. Erstellen Sie die `kustomization.yaml` Verwenden des folgenden Befehls, wobei `<bundle.yaml>` ist `bundle_pre_1_25.yaml` Oder `bundle_post_1_25.yaml` Basierend auf Ihrer Kubernetes-Version

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

- b. Kompilieren Sie das Bündel mit dem folgenden Befehl, wobei `<bundle.yaml>` ist `bundle_pre_1_25.yaml` Oder `bundle_post_1_25.yaml` Basierend auf Ihrer Kubernetes-Version

```
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 Replikasets erstellt wurden.

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



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

### Schritt 4: Erstellen Sie die `TridentOrchestrator` Und `Trident` installieren

Sie können jetzt die erstellen `TridentOrchestrator` Und Installation von `Astra Trident` durchführen. Optional können Sie "[Anpassung der Trident Installation](#)" Verwenden der Attribute im `TridentOrchestrator` Spez.

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

```
kubectl describe torc trident
```

```
Name:          trident
Namespace:
Labels:        <none>
Annotations:   <none>
API Version:  trident.netapp.io/v1
Kind:          TridentOrchestrator
...
Spec:
  Debug:      true
  Namespace:  trident
Status:
  Current Installation Params:
    IPv6:                false
    Autosupport Hostname:
    Autosupport Image:   netapp/trident-autosupport:23.07
    Autosupport Proxy:
    Autosupport Serial Number:
    Debug:                true
    Image Pull Secrets:
    Image Registry:
    k8sTimeout:          30
    Kubelet Dir:         /var/lib/kubelet
    Log Format:           text
    Silence Autosupport: false
    Trident Image:       netapp/trident:23.07.1
  Message:              Trident installed Namespace:
trident
  Status:                Installed
  Version:               v23.07.1
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

Die Installation kann auf verschiedene Weise überprüft werden.

## Wird Verwendet TridentOrchestrator Status

Der Status von `TridentOrchestrator` Gibt an, ob die Installation erfolgreich war und zeigt die installierte Version von Trident an. Während der Installation den Status von `TridentOrchestrator` Änderungen von `Installing` Bis `Installed`. Wenn Sie die beobachten `Failed` Der Status und der Operator kann sich nicht selbst wiederherstellen. ["Prüfen Sie die Protokolle"](#).

Status	Beschreibung
Installation	Der Betreiber installiert damit den Astra Trident <code>TridentOrchestrator</code> CR.
Installiert	Astra Trident wurde erfolgreich installiert.
Deinstallation	Der Betreiber deinstalliert den Astra Trident, denn <code>spec.uninstall=true</code> .
Deinstalliert	Astra Trident ist deinstalliert.
Fehlgeschlagen	Der Bediener konnte nicht installieren, patchen, aktualisieren oder deinstallieren Astra Trident: Der Operator versucht automatisch, eine Wiederherstellung aus diesem Zustand durchzuführen. Wenn dieser Status weiterhin besteht, müssen Sie eine Fehlerbehebung durchführen.
Aktualisierung	Der Bediener aktualisiert eine vorhandene Installation.
Fehler	Der <code>TridentOrchestrator</code> Wird nicht verwendet. Eine weitere bereits Vorhanden.

## Den Status der Pod-Erstellung verwenden

Überprüfen Sie den Status der erstellten Pods, ob die Astra Trident-Installation abgeschlossen wurde:

```
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
```

## Wird Verwendet `tridentctl`

Verwenden Sie können `tridentctl` Um die installierte Version von Astra Trident zu überprüfen.

```
./tridentctl -n trident version

+-----+-----+
| SERVER VERSION | CLIENT VERSION |
+-----+-----+
| 23.07.1       | 23.07.1       |
+-----+-----+
```

## Manuelles Bereitstellen des Trident-Mitarbeiters (Offline-Modus)

Sie können den Trident-Operator manuell implementieren, um Astra Trident zu installieren. Dieser Prozess gilt für Installationen, bei denen die von Astra Trident benötigten Container-Images in einer privaten Registrierung gespeichert werden. Wenn Sie keine private Bildregistrierung besitzen, verwenden Sie das ["Standardimplementierung einsetzen"](#).

### Kritische Informationen zu Astra Trident 23.07

Sie müssen die folgenden wichtigen Informationen über Astra Trident lesen.

#### **Informationen über Astra TriperelT**

- Kubernetes 1.27 wird jetzt in Trident unterstützt. Upgrade von Trident vor dem Upgrade von Kubernetes.
- Astra Trident setzt die Verwendung von Multipathing-Konfiguration in SAN-Umgebungen strikt um und empfiehlt den Nutzen von `find_multipaths: no` In `Multipath.conf` Datei.

Verwendung einer Konfiguration ohne Multipathing oder Verwendung von `find_multipaths: yes` Oder `find_multipaths: smart` Der Wert in der `Multipath.conf`-Datei führt zu Mount-Fehlern. Trident empfiehlt die Verwendung von `find_multipaths: no` Seit der Version 21.07.

## Trident-Operator kann manuell implementiert und Trident installiert werden

Prüfen ["Die Übersicht über die Installation"](#) Um sicherzustellen, dass Sie die Installationsvoraussetzungen erfüllt haben, und die richtige Installationsoption für Ihre Umgebung ausgewählt haben.

### Bevor Sie beginnen

Melden Sie sich beim Linux-Host an, und überprüfen Sie, ob er einen funktionierenden und verwaltet ["Unterstützter Kubernetes-Cluster"](#) Und dass Sie die erforderlichen Berechtigungen haben.





Mit OpenShift, verwenden `oc` Statt `kubectl` In allen folgenden Beispielen, und melden Sie sich als **System:admin** zuerst mit dem Ausführen an `oc login -u system:admin` Oder `oc login -u kube-admin`.

### 1. Überprüfen Sie Ihre Kubernetes Version:

```
kubectl version
```

### 2. Überprüfung der Berechtigungen für Cluster-Administratoren:

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

### 3. Überprüfen Sie, ob Sie einen Pod starten können, der ein Image aus dem Docker Hub verwendet, und ob er das Storage-System über das POD-Netzwerk erreichen kann:

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

## Schritt 1: Laden Sie das Trident Installer-Paket herunter

Das Astra Trident Installationspaket enthält alles, was Sie für die Bereitstellung des Trident-Operators und die Installation von Astra Trident benötigen. Laden Sie die neueste Version des Trident Installationsprogramms herunter und extrahieren Sie sie aus "[Die Sektion Assets auf GitHub](#)".

```
wget https://github.com/NetApp/trident/releases/download/v23.07.1/trident-
installer-23.07.1.tar.gz
tar -xf trident-installer-23.07.1.tar.gz
cd trident-installer
```

## Schritt 2: Erstellen Sie die `TridentOrchestrator` CRD.

Erstellen Sie die `TridentOrchestrator` Benutzerdefinierte Ressourcendefinition (CRD). Sie werden ein erstellen `TridentOrchestrator` Benutzerdefinierte Ressourcen später. Verwenden Sie die entsprechende CRD YAML-Version in `deploy/crds` Um die zu erstellen `TridentOrchestrator` CRD:

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

## Schritt 3: Aktualisieren Sie den Registrierungsort im Operator

In `/deploy/operator.yaml`, Update `image: docker.io/netapp/trident-operator:23.07.1` So geben Sie den Speicherort Ihrer Bildregistrierung an. Ihr "[Trident und CSI-Images](#)" Kann in einer Registrierung

oder in verschiedenen Registern gefunden werden, aber alle CSI-Images müssen sich in derselben Registrierung befinden. Beispiel:

- `image: <your-registry>/trident-operator:23.07.1` Wenn Ihre Bilder alle in einer Registrierung gespeichert sind.
- `image: <your-registry>/netapp/trident-operator:23.07.1` Wenn sich Ihr Trident-Image in einer anderen Registrierung als Ihre CSI-Images befindet.

#### Schritt 4: Implementieren des Trident-Operators

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

- Verwenden Sie für Cluster mit Kubernetes 1.24 oder früheren Versionen `bundle_pre_1_25.yaml`.
- Verwenden Sie für Cluster mit Kubernetes 1.25 oder höher `bundle_post_1_25.yaml`.

#### Bevor Sie beginnen

- Standardmäßig stellt das Trident-Installationsprogramm den Operator in bereit `trident` Namespace. Wenn der `trident` Namespace ist nicht vorhanden, erstellen Sie ihn mit:

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

- Um den Operator in einem anderen Namespace als dem bereitzustellen `trident` Namespace, Update `serviceaccount.yaml`, `clusterrolebinding.yaml` Und `operator.yaml` Und erstellen Sie Ihre Bundle-Datei mit `kustomization.yaml`.
  - a. Erstellen Sie die `kustomization.yaml` Verwenden des folgenden Befehls, wobei `<bundle.yaml>` ist `bundle_pre_1_25.yaml` Oder `bundle_post_1_25.yaml` Basierend auf Ihrer Kubernetes-Version

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

- b. Kompilieren Sie das Bündel mit dem folgenden Befehl, wobei `<bundle.yaml>` ist `bundle_pre_1_25.yaml` Oder `bundle_post_1_25.yaml` Basierend auf Ihrer Kubernetes-Version

```
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 Replikasets erstellt wurden.

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



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

### Schritt 5: Aktualisieren Sie den Speicherort der Bildregistrierung im `TridentOrchestrator`

Ihr "[Trident und CSI-Images](#)" kann in einer Registrierung oder in verschiedenen Registern gefunden werden, aber alle CSI-Images müssen sich in derselben Registrierung befinden. Aktualisierung `deploy/crds/tridentorchestrator_cr.yaml` So fügen Sie zusätzliche Standortspezifikationen basierend auf Ihrer Registrierungskonfiguration hinzu.

#### Bilder in einer Registrierung

```
imageRegistry: "<your-registry>"
autosupportImage: "<your-registry>/trident-autosupport:23.07"
tridentImage: "<your-registry>/trident:23.07.1"
```

#### Bilder in verschiedenen Registern

Sie müssen anhängen `sig-storage` Bis zum `imageRegistry` Um unterschiedliche Registrierungsstandorte zu verwenden.

```
imageRegistry: "<your-registry>/sig-storage"
autosupportImage: "<your-registry>/netapp/trident-autosupport:23.07"
tridentImage: "<your-registry>/netapp/trident:23.07.1"
```

### Schritt 6: Erstellen Sie die `TridentOrchestrator` Und Trident installieren

Sie können jetzt die erstellen `TridentOrchestrator` Und Installation von Astra Trident durchführen. Optional können Sie weiter "[Anpassung der Trident Installation](#)" verwenden der Attribute im `TridentOrchestrator` Spez. Das folgende Beispiel zeigt eine Installation, bei der sich Trident- und CSI-Bilder in verschiedenen Registern 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>/netapp/trident-autosupport:23.07
  Debug:              true
  Image Registry:    <your-registry>/sig-storage
  Namespace:         trident
  Trident Image:     <your-registry>/netapp/trident:23.07.1
Status:
  Current Installation Params:
    IPv6:              false
    Autosupport Hostname:
    Autosupport Image: <your-registry>/netapp/trident-
autosupport:23.07
    Autosupport Proxy:
    Autosupport Serial Number:
    Debug:              true
    Http Request Timeout: 90s
    Image Pull Secrets:
    Image Registry:    <your-registry>/sig-storage
    k8sTimeout:        30
    Kubelet Dir:       /var/lib/kubelet
    Log Format:         text
    Probe Port:        17546
    Silence Autosupport: false
    Trident Image:     <your-registry>/netapp/trident:23.07.1
  Message:            Trident installed
  Namespace:          trident
  Status:              Installed
  Version:             v23.07.1
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

Die Installation kann auf verschiedene Weise überprüft werden.

### Wird Verwendet `TridentOrchestrator` Status

Der Status von `TridentOrchestrator` Gibt an, ob die Installation erfolgreich war und zeigt die installierte Version von Trident an. Während der Installation den Status von `TridentOrchestrator` Änderungen von `Installing` Bis `Installed`. Wenn Sie die beobachten `Failed` Der Status und der Operator kann sich nicht selbst wiederherstellen. "[Prüfen Sie die Protokolle](#)".

Status	Beschreibung
Installation	Der Betreiber installiert damit den Astra Trident <code>TridentOrchestrator</code> CR.
Installiert	Astra Trident wurde erfolgreich installiert.
Deinstallation	Der Betreiber deinstalliert den Astra Trident, denn <code>spec.uninstall=true</code> .
Deinstalliert	Astra Trident ist deinstalliert.
Fehlgeschlagen	Der Bediener konnte nicht installieren, patchen, aktualisieren oder deinstallieren Astra Trident: Der Operator versucht automatisch, eine Wiederherstellung aus diesem Zustand durchzuführen. Wenn dieser Status weiterhin besteht, müssen Sie eine Fehlerbehebung durchführen.
Aktualisierung	Der Bediener aktualisiert eine vorhandene Installation.
Fehler	Der <code>TridentOrchestrator</code> Wird nicht verwendet. Eine weitere bereits Vorhanden.

### Den Status der Pod-Erstellung verwenden

Überprüfen Sie den Status der erstellten Pods, ob die Astra Trident-Installation abgeschlossen wurde:

```
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

### Wird Verwendet `tridentctl`

Verwenden Sie können `tridentctl` Um die installierte Version von Astra Trident zu überprüfen.

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

```
+-----+-----+
| SERVER VERSION | CLIENT VERSION |
+-----+-----+
| 23.07.1       | 23.07.1       |
+-----+-----+
```

## Trident Operator mit Helm (Standard-Modus) implementieren

Sie können den Trident-Operator implementieren und Astra Trident mithilfe von Helm installieren. Dieser Prozess gilt für Installationen, bei denen die von Astra Trident benötigten Container-Images nicht in einer privaten Registrierung gespeichert werden. Wenn Sie über eine private Bildregistrierung verfügen, verwenden Sie das ["Prozess für Offline-Implementierung"](#).

### Kritische Informationen zu Astra Trident 23.07.1

Sie müssen die folgenden wichtigen Informationen über Astra Trident lesen.

## <strong> Informationen über Astra Trident </strong>

- Kubernetes 1.27 wird jetzt in Trident unterstützt. Upgrade von Trident vor dem Upgrade von Kubernetes.
- Astra Trident setzt die Verwendung von Multipathing-Konfiguration in SAN-Umgebungen strikt um und empfiehlt den Nutzen von `find_multipaths: no` In Multipath.conf Datei.

Verwendung einer Konfiguration ohne Multipathing oder Verwendung von `find_multipaths: yes` Oder `find_multipaths: smart` Der Wert in der Multipath.conf-Datei führt zu Mount-Fehlern. Trident empfiehlt die Verwendung von `find_multipaths: no` Seit der Version 21.07.

## Setzen Sie den Trident-Operator ein und installieren Sie Astra Trident mit Helm

Verwendung von Trident "[Steuerruderdiagramm](#)" Sie können den Trident Operator implementieren und Trident in einem Schritt installieren.

Prüfen "[Die Übersicht über die Installation](#)" Um sicherzustellen, dass Sie die Installationsvoraussetzungen erfüllt haben, und die richtige Installationsoption für Ihre Umgebung ausgewählt haben.

### Bevor Sie beginnen

Zusätzlich zum "[Voraussetzungen für die Implementierung](#)" Die Sie benötigen "[Helm Version 3](#)".

### Schritte

1. Fügen Sie das Helm Repository von Astra Trident hinzu:

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

2. Nutzung `helm install` Und geben Sie einen Namen für Ihre Bereitstellung an, wie im folgenden Beispiel, wo `23.07.1` Ist die Version des Astra Trident, die Sie installieren.

```
helm install <name> netapp-trident/trident-operator --version 23.07.1  
--create-namespace --namespace <trident-namespace>
```



Wenn Sie bereits einen Namespace für Trident erstellt haben, wird der `--create-namespace` Parameter erstellt keinen zusätzlichen Namespace.

Verwenden Sie können `helm list` So prüfen Sie Installationsdetails wie Name, Namespace, Diagramm, Status, App-Version, Und Revisionsnummer.

## Konfigurationsdaten während der Installation übergeben

Während der Installation gibt es zwei Möglichkeiten, die Konfigurationsdaten zu übergeben:

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

Um beispielsweise den Standardwert von `debug` zu ändern, Ausführen Sie das folgende `--set` Befehl wo `23.07.1` Ist die Version von Astra Trident, die Sie installieren:

```
helm install <name> netapp-trident/trident-operator --version 23.07.1
--create-namespace --namespace trident --set tridentDebug=true
```

## Konfigurationsoptionen

Diese Tabelle und die `values.yaml` Datei, die Teil des Helm-Diagramms ist, enthält die Liste der Schlüssel und ihre Standardwerte.

Option	Beschreibung	Standard
<code>nodeSelector</code>	Node-Etiketten für Pod-Zuweisung	
<code>podAnnotations</code>	Pod-Anmerkungen	
<code>deploymentAnnotations</code>	Anmerkungen zur Bereitstellung	
<code>tolerations</code>	Toleranzen für Pod-Zuweisung	
<code>affinity</code>	Affinität für Pod-Zuweisung	
<code>tridentControllerPluginNodeSelector</code>	Zusätzliche Node-Auswahl für Pods Siehe <a href="#">Allgemeines zu Controller-Pods und Node-Pods</a> Entsprechende Details.	
<code>tridentControllerPluginTolerations</code>	Überschreibt Kubernetes-Toleranzen für Pods. Siehe <a href="#">Allgemeines zu Controller-Pods und Node-Pods</a> Entsprechende Details.	
<code>tridentNodePluginNodeSelector</code>	Zusätzliche Node-Auswahl für Pods Siehe <a href="#">Allgemeines zu Controller-Pods und Node-Pods</a> Entsprechende Details.	
<code>tridentNodePluginTolerations</code>	Überschreibt Kubernetes-Toleranzen für Pods. Siehe <a href="#">Allgemeines zu Controller-Pods und Node-Pods</a> Entsprechende Details.	



Option	Beschreibung	Standard
imageRegistry	Identifiziert die Registrierung für den <code>trident-operator</code> , <code>trident</code> , Und andere Bilder. Lassen Sie das Feld leer, um die Standardeinstellung zu übernehmen.	“
imagePullPolicy	Legt die Richtlinie zum Abziehen von Bildern für den fest <code>trident-operator</code> .	IfNotPresent
imagePullSecrets	Legt die Abzugsgeheimnisse für das Bild fest <code>trident-operator</code> , <code>trident</code> , Und andere Bilder.	
kubeletDir	Ermöglicht das Überschreiben der Hostposition des internen Status von kubelet.	“/var/lib/kubelet“
operatorLogLevel	Ermöglicht die Einstellung der Protokollebene des Trident-Operators auf: <code>trace</code> , <code>debug</code> , <code>info</code> , <code>warn</code> , <code>error</code> , Oder <code>fatal</code> .	"info"
operatorDebug	Ermöglicht es, die Protokollebene des Trident-Operators auf Debug zu setzen.	true
operatorImage	Ermöglicht die vollständige Überschreibung des Bildes für <code>trident-operator</code> .	“
operatorImageTag	Ermöglicht das Überschreiben des Tags des <code>trident-operator</code> Bild:	“
tridentIPv6	Ermöglicht die Aktivierung von Astra Trident in IPv6-Clustern.	false
tridentK8sTimeout	Setzt das standardmäßige 30-Sekunden-Zeitlimit für die meisten Kubernetes-API-Vorgänge außer Kraft (wenn nicht Null, in Sekunden).	0
tridentHttpRequestTimeout	Setzt das standardmäßige 90-Sekunden-Timeout für die HTTP-Anforderungen mit außer Kraft <code>0s</code> Ist eine unendliche Dauer für das Timeout. Negative Werte sind nicht zulässig.	"90s"
tridentSilenceAutosupport	Ermöglicht die Deaktivierung von regelmäßigen AutoSupport Berichten für Astra Trident.	false

Option	Beschreibung	Standard
tridentAutosupportImageTag	Ermöglicht das Überschreiben des Tags des Images für den Astra Trident AutoSupport-Container.	<version>
tridentAutosupportProxy	Der Astra Trident AutoSupport Container kann über einen HTTP-Proxy nach Hause telefonieren.	“
tridentLogFormat	Legt das Astra Trident Protokollierungsformat fest (text Oder json).	"text"
tridentDisableAuditLog	Deaktiviert den Astra Trident Audit-Logger.	true
tridentLogLevel	Ermöglicht die Festlegung der Protokollebene von Astra Trident auf: trace, debug, info, warn, error, Oder fatal.	"info"
tridentDebug	Ermöglicht das Festlegen der Protokollebene für Astra Trident debug.	false
tridentLogWorkflows	Ermöglicht die Aktivierung bestimmter Astra Trident Workflows für die Trace-Protokollierung oder Protokollunterdrückung.	“
tridentLogLayers	Ermöglicht die Aktivierung bestimmter Astra Trident-Ebenen für die Trace-Protokollierung oder Protokollunterdrückung.	“
tridentImage	Ermöglicht die vollständige Überschreibung des Images für Astra Trident.	“
tridentImageTag	Ermöglicht das Überschreiben des Tags des Images für Astra Trident.	“
tridentProbePort	Ermöglicht das Überschreiben des Standardports, der für Kubernetes Liveness/Readiness-Sonden verwendet wird.	“
windows	Ermöglicht die Installation von Astra Trident auf einem Windows Worker-Node.	false
enableForceDetach	Ermöglicht die Aktivierung der Funktion zum Abtrennen erzwingen.	false
excludePodSecurityPolicy	Schließt die Sicherheitsrichtlinie des Operator POD von der Erstellung aus.	false

## Allgemeines zu Controller-Pods und Node-Pods

Astra Trident wird als einzelner Controller-Pod ausgeführt sowie als Node-Pod auf jedem Worker-Node im Cluster. Der Node Pod muss auf jedem Host ausgeführt werden, auf dem Sie ein Astra Trident Volume mounten möchten.

Kubernetes "[Knotenauswahl](#)" Und "[Toleranzen und Verfleckungen](#)" Werden verwendet, um die Ausführung eines Pod auf einem bestimmten oder bevorzugten Node einzuschränken. Verwenden von `ControllerPlugin`` und `NodePlugin`, Sie können Bedingungen und Überschreibungen festlegen.

- Das Controller-Plug-in übernimmt Volume-Bereitstellung und -Management, beispielsweise Snapshots und Größenanpassungen.
- Das Node-Plug-in verarbeitet das Verbinden des Speichers mit dem Node.

## Trident-Operator mit Helm (Offline-Modus) implementieren

Sie können den Trident-Operator implementieren und Astra Trident mithilfe von Helm installieren. Dieser Prozess gilt für Installationen, bei denen die von Astra Trident benötigten Container-Images in einer privaten Registrierung gespeichert werden. Wenn Sie keine private Bildregistrierung besitzen, verwenden Sie das "[Standardimplementierung einsetzen](#)".

### Kritische Informationen zu Astra Trident 23.07

Sie müssen die folgenden wichtigen Informationen über Astra Trident lesen.

#### **Informationen über Astra TriperelT**

- Kubernetes 1.27 wird jetzt in Trident unterstützt. Upgrade von Trident vor dem Upgrade von Kubernetes.
- Astra Trident setzt die Verwendung von Multipathing-Konfiguration in SAN-Umgebungen strikt um und empfiehlt den Nutzen von `find_multipaths: no` In Multipath.conf Datei.

Verwendung einer Konfiguration ohne Multipathing oder Verwendung von `find_multipaths: yes` Oder `find_multipaths: smart` Der Wert in der Multipath.conf-Datei führt zu Mount-Fehlern. Trident empfiehlt die Verwendung von `find_multipaths: no` Seit der Version 21.07.

## Setzen Sie den Trident-Operator ein und installieren Sie Astra Trident mit Helm

Verwendung von Trident "[Steuerruderdiagramm](#)" Sie können den Trident Operator implementieren und Trident in einem Schritt installieren.

Prüfen "[Die Übersicht über die Installation](#)" Um sicherzustellen, dass Sie die Installationsvoraussetzungen erfüllt haben, und die richtige Installationsoption für Ihre Umgebung ausgewählt haben.

### Bevor Sie beginnen

Zusätzlich zum "[Voraussetzungen für die Implementierung](#)" Die Sie benötigen "[Helm Version 3](#)".

### Schritte

1. Fügen Sie das Helm Repository von Astra Trident hinzu:

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

2. Nutzung `helm install` Und geben Sie einen Namen für die Bereitstellung und den Speicherort der Image-Registrierung an. Ihr "Trident und CSI-Images" Kann in einer Registrierung oder in verschiedenen Registern gefunden werden, aber alle CSI-Images müssen sich in derselben Registrierung befinden. In den Beispielen: 23.07.1 Ist die Version des Astra Trident, die Sie installieren.

### Bilder in einer Registrierung

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

### Bilder in verschiedenen Registern

Sie müssen anhängen `sig-storage` Bis zum `imageRegistry` Um unterschiedliche Registrierungsstandorte zu verwenden.

```
helm install <name> netapp-trident/trident-operator --version  
23.07.1 --set imageRegistry=<your-registry>/sig-storage --set  
operatorImage=<your-registry>/netapp/trident-operator:23.07.1 --set  
tridentAutosupportImage=<your-registry>/netapp/trident-  
autosupport:23.07 --set tridentImage=<your-  
registry>/netapp/trident:23.07.1 --create-namespace --namespace  
<trident-namespace>
```



Wenn Sie bereits einen Namespace für Trident erstellt haben, wird der `--create-namespace` Parameter erstellt keinen zusätzlichen Namespace.

Verwenden Sie können `helm list` So prüfen Sie Installationsdetails wie Name, Namespace, Diagramm, Status, App-Version, Und Revisionsnummer.

## Konfigurationsdaten während der Installation übergeben

Während der Installation gibt es zwei Möglichkeiten, die Konfigurationsdaten 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 rechteste Datei hat Vorrang.

Option	Beschreibung
--set	Geben Sie Überschreibungen in der Befehlszeile an.

Um beispielsweise den Standardwert von `debug` zu ändern, Ausführen Sie das folgende `--set` Befehl wo `23.07.1` Ist die Version von Astra Trident, die Sie installieren:

```
helm install <name> netapp-trident/trident-operator --version 23.07.1
--create-namespace --namespace trident --set tridentDebug=true
```

## Konfigurationsoptionen

Diese Tabelle und die `values.yaml` Datei, die Teil des Helm-Diagramms ist, enthält die Liste der Schlüssel und ihre Standardwerte.

Option	Beschreibung	Standard
<code>nodeSelector</code>	Node-Etiketten für Pod-Zuweisung	
<code>podAnnotations</code>	Pod-Anmerkungen	
<code>deploymentAnnotations</code>	Anmerkungen zur Bereitstellung	
<code>tolerations</code>	Toleranzen für Pod-Zuweisung	
<code>affinity</code>	Affinität für Pod-Zuweisung	
<code>tridentControllerPluginNodeSelector</code>	Zusätzliche Node-Auswahl für Pods Siehe <a href="#">Allgemeines zu Controller-Pods und Node-Pods</a> Entsprechende Details.	
<code>tridentControllerPluginTolerations</code>	Überschreibt Kubernetes-Toleranzen für Pods. Siehe <a href="#">Allgemeines zu Controller-Pods und Node-Pods</a> Entsprechende Details.	
<code>tridentNodePluginNodeSelector</code>	Zusätzliche Node-Auswahl für Pods Siehe <a href="#">Allgemeines zu Controller-Pods und Node-Pods</a> Entsprechende Details.	
<code>tridentNodePluginTolerations</code>	Überschreibt Kubernetes-Toleranzen für Pods. Siehe <a href="#">Allgemeines zu Controller-Pods und Node-Pods</a> Entsprechende Details.	

Option	Beschreibung	Standard
imageRegistry	Identifiziert die Registrierung für den <code>trident-operator</code> , <code>trident</code> , Und andere Bilder. Lassen Sie das Feld leer, um die Standardeinstellung zu übernehmen.	“
imagePullPolicy	Legt die Richtlinie zum Abziehen von Bildern für den fest <code>trident-operator</code> .	IfNotPresent
imagePullSecrets	Legt die Abzugsgeheimnisse für das Bild fest <code>trident-operator</code> , <code>trident</code> , Und andere Bilder.	
kubeletDir	Ermöglicht das Überschreiben der Hostposition des internen Status von kubelet.	“/var/lib/kubelet“
operatorLogLevel	Ermöglicht die Einstellung der Protokollebene des Trident-Operators auf: <code>trace</code> , <code>debug</code> , <code>info</code> , <code>warn</code> , <code>error</code> , Oder <code>fatal</code> .	"info"
operatorDebug	Ermöglicht es, die Protokollebene des Trident-Operators auf Debug zu setzen.	true
operatorImage	Ermöglicht die vollständige Überschreibung des Bildes für <code>trident-operator</code> .	“
operatorImageTag	Ermöglicht das Überschreiben des Tags des <code>trident-operator</code> Bild:	“
tridentIPv6	Ermöglicht die Aktivierung von Astra Trident in IPv6-Clustern.	false
tridentK8sTimeout	Setzt das standardmäßige 30-Sekunden-Zeitlimit für die meisten Kubernetes-API-Vorgänge außer Kraft (wenn nicht Null, in Sekunden).	0
tridentHttpRequestTimeout	Setzt das standardmäßige 90-Sekunden-Timeout für die HTTP-Anforderungen mit außer Kraft 0s Ist eine unendliche Dauer für das Timeout. Negative Werte sind nicht zulässig.	"90s"
tridentSilenceAutosupport	Ermöglicht die Deaktivierung von regelmäßigen AutoSupport Berichten für Astra Trident.	false

Option	Beschreibung	Standard
tridentAutosupportImageTag	Ermöglicht das Überschreiben des Tags des Images für den Astra Trident AutoSupport-Container.	<version>
tridentAutosupportProxy	Der Astra Trident AutoSupport Container kann über einen HTTP-Proxy nach Hause telefonieren.	“
tridentLogFormat	Legt das Astra Trident Protokollierungsformat fest (text Oder json).	"text"
tridentDisableAuditLog	Deaktiviert den Astra Trident Audit-Logger.	true
tridentLogLevel	Ermöglicht die Festlegung der Protokollebene von Astra Trident auf: trace, debug, info, warn, error, Oder fatal.	"info"
tridentDebug	Ermöglicht das Festlegen der Protokollebene für Astra Trident debug.	false
tridentLogWorkflows	Ermöglicht die Aktivierung bestimmter Astra Trident Workflows für die Trace-Protokollierung oder Protokollunterdrückung.	“
tridentLogLayers	Ermöglicht die Aktivierung bestimmter Astra Trident-Ebenen für die Trace-Protokollierung oder Protokollunterdrückung.	“
tridentImage	Ermöglicht die vollständige Überschreibung des Images für Astra Trident.	“
tridentImageTag	Ermöglicht das Überschreiben des Tags des Images für Astra Trident.	“
tridentProbePort	Ermöglicht das Überschreiben des Standardports, der für Kubernetes Liveness/Readiness-Sonden verwendet wird.	“
windows	Ermöglicht die Installation von Astra Trident auf einem Windows Worker-Node.	false
enableForceDetach	Ermöglicht die Aktivierung der Funktion zum Abtrennen erzwingen.	false
excludePodSecurityPolicy	Schließt die Sicherheitsrichtlinie des Operator POD von der Erstellung aus.	false

## Wie es weiter geht

# Anpassen der Trident Operator-Installation

Über den Trident-Operator können Sie die Astra Trident-Installation anhand der Attribute im anpassen `TridentOrchestrator` Spez. Wenn Sie die Installation über die von Ihnen gewünschte hinaus anpassen möchten `TridentOrchestrator` Argumente erlauben, verwenden Sie `tridentctl` Um benutzerdefinierte YAML-Manifeste zu erzeugen, die bei Bedarf geändert werden sollen.

## Allgemeines zu Controller-Pods und Node-Pods

Astra Trident wird als einzelner Controller-Pod ausgeführt sowie als Node-Pod auf jedem Worker-Node im Cluster. Der Node Pod muss auf jedem Host ausgeführt werden, auf dem Sie ein Astra Trident Volume mounten möchten.

Kubernetes "[Knotenauswahl](#)" Und "[Toleranzen und Verfleckungen](#)" Werden verwendet, um die Ausführung eines Pod auf einem bestimmten oder bevorzugten Node einzuschränken. Verwenden von `ControllerPlugin` und `NodePlugin`, Sie können Bedingungen und Überschreibungen festlegen.

- Das Controller-Plug-in übernimmt Volume-Bereitstellung und -Management, beispielsweise Snapshots und Größenanpassungen.
- Das Node-Plug-in verarbeitet das Verbinden des Speichers mit dem Node.

## Konfigurationsoptionen



`spec.namespace` Ist in angegeben `TridentOrchestrator` Um den Namespace zu kennzeichnen, in dem Astra Trident installiert ist. Dieser Parameter **kann nicht aktualisiert werden, nachdem Astra Trident installiert wurde**. Der Versuch, dies zu tun, bewirkt das `TridentOrchestrator` Status zu ändern in `Failed`. Astra Trident ist nicht für die Migration auf Namespaces vorgesehen.

Diese Tabelle enthält Einzelheiten `TridentOrchestrator` Attribute.

Parameter	Beschreibung	Standard
<code>namespace</code>	Namespace für die Installation von Astra Trident in	„Standard“
<code>debug</code>	Aktivieren Sie das Debugging für Astra Trident	Falsch
<code>enableForceDetach</code>	<code>ontap-san</code> Und <code>ontap-san-economy</code> Nur.  Arbeitet mit Kubernetes Non-Graceful Node Shutdown (NGNS), um Clusteradministratoren die Möglichkeit zu geben, Workloads mit gemounteten Volumes sicher auf neue Nodes zu migrieren, sollte ein Node fehlerhaft werden.  <b>Dies ist ein experimentelles Feature in 23.04.</b> Siehe <a href="#">Details zum Ablösen von Krafteinwirkung</a> Wichtige Hinweise.	<code>false</code>



Parameter	Beschreibung	Standard
windows	Einstellung auf <code>true</code> Ermöglicht die Installation auf Windows Worker-Knoten.	Falsch
IPv6	Installieren Sie Astra Trident über IPv6	Falsch
k8sTimeout	Zeitüberschreitung für Kubernetes-Betrieb	30 Sek.
silenceAutosupport	Senden Sie keine AutoSupport Bundles an NetApp Automatisch	Falsch
autosupportImage	Das Container-Image für AutoSupport Telemetrie	„netapp/Trident-AutoSupport:23.07“
autosupportProxy	Die Adresse/den Port eines Proxys zum Senden von AutoSupport Telemetrie	"<a href="http://proxy.example.com:8888" class="bare">http://proxy.example.com:8888"</a>
uninstall	Eine Flagge, die zum Deinstallieren von Astra Trident verwendet wird	Falsch
logFormat	Astra Trident Protokollformat zur Verwendung [Text, json]	„Text“
tridentImage	Astra Trident-Image zu installieren	„netapp/Trident:23.07“
imageRegistry	Pfad zur internen Registrierung des Formats <registry FQDN>[:port] [/subpath]	„K8s.gcr.io/SIG-Storage“ (K8s 1.19+) Oder „Quay.io/k8scsi“
kubeletDir	Pfad zum kubelet-Verzeichnis auf dem Host	„/var/lib/kubelet“
wipeout	Eine Liste der zu löschenden Ressourcen, um eine vollständige Entfernung von durchzuführen Astra Trident	
imagePullSecrets	Secrets, um Bilder aus einer internen Registrierung zu ziehen	
imagePullPolicy	Legt die BildPull-Richtlinie für den Trident-Operator fest. Gültige Werte sind:  Always Um immer das Bild zu ziehen.  IfNotPresent Nur wenn das Image nicht auf dem Node vorhanden ist, soll das Image kopiert werden.  Never Nie das Bild ziehen.	IfNotPresent
controllerPluginNodeSelector	Zusätzliche Node-Auswahl für Pods Entspricht dem Format <code>pod.spec.nodeSelector</code> .	Kein Standard; optional
controllerPluginTolerations	Überschreibt Kubernetes-Toleranzen für Pods. Entspricht dem gleichen Format wie <code>pod.spec.Tolerations</code> .	Kein Standard; optional

Parameter	Beschreibung	Standard
<code>nodePluginNodeSelector</code>	Zusätzliche Node-Auswahl für Pods Entspricht dem Format <code>pod.spec.nodeSelector</code> .	Kein Standard; optional
<code>nodePluginTolerations</code>	Überschreibt Kubernetes-Toleranzen für Pods. Entspricht dem gleichen Format wie <code>pod.spec.Tolerations</code> .	Kein Standard; optional



Weitere Informationen zum Formatieren von Pod-Parametern finden Sie unter ["Pods werden Nodes zugewiesen"](#).

## Details zum Ablösen von Krafteinwirkung

Trennung erzwingen ist für verfügbar `ontap-san` Und `ontap-san-economy` Nur. Vor der Aktivierung von Force Trennen muss das nicht-anmutige Herunterfahren des Node (NGNS) auf dem Kubernetes-Cluster aktiviert sein. Weitere Informationen finden Sie unter ["Kubernetes: Nicht ordnungsgemäßes Herunterfahren von Nodes"](#).



Da Astra Trident auf Kubernetes NGNS basiert, entfernen Sie nicht `out-of-service` Bleibt bei einem Node in einem unzulässigen Zustand, bis alle nicht tolerierbaren Workloads neu geplant werden. Das rücksichtslose Anwenden oder Entfernen der Schein kann den Schutz der Back-End-Daten gefährden.

Wenn der Kubernetes-Cluster-Administrator den angewendet hat `node.kubernetes.io/out-of-service=nodeshutdown:NoExecute` Taint to the Node and `enableForceDetach` Ist auf festgelegt ``true`` Astra Trident bestimmt den Node-Status und:

1. Beenden Sie den Back-End-I/O-Zugriff für Volumes, die auf diesem Node gemountet sind.
2. Markieren Sie das Astra Trident Node-Objekt als `dirty` (Nicht sicher für neue Publikationen).



Der Trident-Controller lehnt neue Anfragen zu veröffentlichten Volumes ab, bis der Node neu qualifiziert wird (nachdem er als markiert wurde) `dirty` Durch den Trident Node POD. Alle Workloads, die mit einer gemounteten PVC geplant sind (selbst nachdem der Cluster-Node funktionsfähig und bereit ist), werden erst akzeptiert, wenn Astra Trident den Node überprüfen kann `clean` (Sicher für neue Publikationen).

Wenn der Zustand der Nodes wiederhergestellt und die Wartung entfernt wird, übernimmt Astra Trident folgende Aufgaben:

1. Veraltete veröffentlichte Pfade auf dem Node identifizieren und bereinigen.
2. Wenn sich der Node in einem befindet `cleanable` Status (die `ServiceTaint` wurde entfernt, und der Node befindet sich in `Ready` Status) und alle veralteten, veröffentlichten Pfade sind sauber. Astra Trident übermittelt den Node als neu `clean` Und neue veröffentlichte Volumes auf dem Knoten zulassen.

## Beispielkonfigurationen

Sie können die oben genannten Attribute beim Definieren verwenden `TridentOrchestrator` Um die Installation anzupassen.

## Beispiel 1: Grundlegende benutzerdefinierte Konfiguration

Dies ist ein Beispiel für eine benutzerdefinierte Grundkonfiguration.

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

## Beispiel 2: Implementierung mit Node-Auswahl

Dieses Beispiel veranschaulicht die Implementierung von Trident mit Node-Selektoren:

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

### Beispiel 3: Bereitstellung auf Windows Worker-Nodes

Dieses Beispiel zeigt die Bereitstellung auf einem Windows Worker-Knoten.

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

## Copyright-Informationen

Copyright © 2024 NetApp. Alle Rechte vorbehalten. Gedruckt in den USA. Dieses urheberrechtlich geschützte Dokument darf ohne die vorherige schriftliche Genehmigung des Urheberrechtinhabers 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.