Installation über den Trident Operator

Astra Trident

NetApp
April 16, 2024

Inhalt

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".

Entscheidende Informationen zu Astra Trident 23.01

Sie müssen die folgenden wichtigen Informationen über Astra Trident lesen.

** Informationen über Astra TripereIT **

- Kubernetes 1.26 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 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/v23.01.1/trident-
installer-23.01.1.tar.gz
tar -xf trident-installer-23.01.1.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 älter `bundle_pre_1_25.yaml`.

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

Das Trident-Installationsprogramm implementiert den Operator im `trident` Namespace. Wenn der `trident` Der Namespace ist nicht vorhanden, verwenden Sie `kubectl apply -f deploy/namespace.yaml` Um sie zu erstellen.

Schritte

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

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

2. Überprüfen Sie, ob der Bediener bereitgestellt wurde.

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

NAME                READY   UP-TO-DATE   AVAILABLE   AGE
trident-operator    1/1     1            1           3m
```

> 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:23.01
    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:23.01.1
  Message:                  Trident installed  Namespace:
trident
  Status:                   Installed
  Version:                  v23.01.1
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 `TridentOrchestrator` CR.
Installiert	Astra Trident wurde erfolgreich installiert.
Deinstallation	Der Betreiber deinstalliert den Astra Trident, denn `spec.uninstall=true`.
Deinstalliert	Astra Trident ist deinstalliert.
Fehlgeschlagen	Der Operator konnte Astra Trident nicht installieren, patchen, aktualisieren oder deinstallieren; der Operator versucht automatisch, aus diesem Zustand wiederherzustellen. Wenn dieser Status weiterhin besteht, müssen Sie eine Fehlerbehebung durchführen.
Aktualisierung	Der Bediener aktualisiert eine vorhandene Installation.
Fehler	Der `TridentOrchestrator` Wird nicht verwendet. Eine weitere ist 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 |
+----------------+----------------+
| 23.01.1        | 23.01.1        |
+----------------+----------------+
```

Wie es weiter geht

Das ist jetzt möglich "Erstellen Sie ein Back-End und eine Storage-Klasse, stellen Sie ein Volume bereit und mounten Sie das Volume in einem POD".

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".

Entscheidende Informationen zu Astra Trident 23.01

Sie müssen die folgenden wichtigen Informationen über Astra Trident lesen.

** Informationen über Astra TripereIT **

- Kubernetes 1.26 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/v23.01.1/trident-
installer-23.01.1.tar.gz
tar -xf trident-installer-23.01.1.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:23.01.1` 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:23.01.1` Wenn Ihre Bilder alle in einer Registrierung gespeichert sind.
- `image: <your-registry>/netapp/trident-operator:23.01.1` Wenn sich Ihr Trident-Image in einer anderen Registrierung als Ihre CSI-Images befindet.

Schritt 4: Implementieren des Trident-Operators

Das Trident-Installationsprogramm implementiert den Operator im `trident` Namespace. Wenn der `trident` Der Namespace ist nicht vorhanden, verwenden Sie `kubectl apply -f deploy/namespace.yaml` Um sie zu erstellen.

Um den Operator in einem anderen Namespace als dem bereitzustellen `trident` Namespace, Update `serviceaccount.yaml`, `clusterrolebinding.yaml` Und `operator.yaml` Vor der Bereitstellung des Bedieners.

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

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

> 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 älter `bundle_pre_1_25.yaml`.
> - Verwenden Sie für Cluster mit Kubernetes 1.25 oder höher `bundle_post_1_25.yaml`.

2. Überprüfen Sie, ob der Bediener bereitgestellt wurde.

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

NAME                READY   UP-TO-DATE   AVAILABLE   AGE
trident-operator    1/1     1            1           3m
```

> 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:23.01"
tridentImage: "<your-registry>/trident:23.01.1"
```

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:23.01"
tridentImage: "<your-registry>/netapp/trident:23.01.1"
```

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.

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

kubectl describe torc trident

Name:        trident
Namespace:
Labels:      <none>
Annotations: <none>
API Version: trident.netapp.io/v1
Kind:        TridentOrchestrator
...
Spec:
  Autosupport Image:  <your-registry>/netapp/trident-autosupport:23.01
  Debug:              true
  Image Registry:     <your-registry>/sig-storage
  Namespace:          trident
  Trident Image:      <your-registry>/netapp/trident:23.01.1
Status:
  Current Installation Params:
    IPv6:                       false
    Autosupport Hostname:
    Autosupport Image:          <your-registry>/netapp/trident-
autosupport:23.01
    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:23.01.1
  Message:                Trident installed
  Namespace:              trident
  Status:                 Installed
  Version:                v23.01.1
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 `TridentOrchestrator` CR.
Installiert	Astra Trident wurde erfolgreich installiert.
Deinstallation	Der Betreiber deinstalliert den Astra Trident, denn `spec.uninstall=true`.
Deinstalliert	Astra Trident ist deinstalliert.
Fehlgeschlagen	Der Operator konnte Astra Trident nicht installieren, patchen, aktualisieren oder deinstallieren; der Operator versucht automatisch, aus diesem Zustand wiederherzustellen. Wenn dieser Status weiterhin besteht, müssen Sie eine Fehlerbehebung durchführen.
Aktualisierung	Der Bediener aktualisiert eine vorhandene Installation.
Fehler	Der `TridentOrchestrator` Wird nicht verwendet. Eine weitere ist 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 |
+----------------+----------------+
| 23.01.1        | 23.01.1        |
+----------------+----------------+
```

Wie es weiter geht

Das ist jetzt möglich "Erstellen Sie ein Back-End und eine Storage-Klasse, stellen Sie ein Volume bereit und mounten Sie das Volume in einem POD".

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".

Entscheidende Informationen zu Astra Trident 23.01

Sie müssen die folgenden wichtigen Informationen über Astra Trident lesen.

** Informationen über Astra TripereIT **

- Kubernetes 1.26 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 `23.01.1` Ist die Version des Astra Trident, die Sie installieren.

```
helm install <name> netapp-trident/trident-operator --version 23.01.1
--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
`--values` (Oder `-f`)	Geben Sie eine YAML-Datei mit Überschreibungen an. Dies kann mehrfach angegeben werden, und die rechteste Datei hat Vorrang.
`--set`	Geben Sie Überschreibungen in der Befehlszeile an.

Um beispielsweise den Standardwert von zu ändern `debug`, Ausführen Sie das folgende `--set` Befehl wo `23.01.1` Ist die Version von Astra Trident, die Sie installieren:

```
helm install <name> netapp-trident/trident-operator --version 23.01.1
--create-namespace --namespace --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
`nodeSelector`	Node-Etiketten für Pod-Zuweisung	
`podAnnotations`	Pod-Anmerkungen	
`deploymentAnnotations`	Anmerkungen zur Bereitstellung	
`tolerations`	Toleranzen für Pod-Zuweisung	
`affinity`	Affinität für Pod-Zuweisung	
`tridentControllerPluginNodeSelector`	Zusätzliche Node-Auswahl für Pods Siehe Allgemeines zu Controller-Pods und Node-Pods Entsprechende Details.	
`tridentControllerPluginTolerations`	Überschreibt Kubernetes-Toleranzen für Pods. Siehe Allgemeines zu Controller-Pods und Node-Pods Entsprechende Details.	
`tridentNodePluginNodeSelector`	Zusätzliche Node-Auswahl für Pods Siehe Allgemeines zu Controller-Pods und Node-Pods Entsprechende Details.	
`tridentNodePluginTolerations`	Überschreibt Kubernetes-Toleranzen für Pods. Siehe Allgemeines zu Controller-Pods und Node-Pods Entsprechende Details.	

Option	Beschreibung	Standard
`imageRegistry`	Identifiziert die Registrierung für den `trident-operator`, `trident`, Und andere Bilder. Lassen Sie das Feld leer, um die Standardeinstellung zu übernehmen.	„“
`imagePullPolicy`	Legt die Richtlinie zum Abziehen von Bildern für den fest `trident-operator`.	`IfNotPresent`
`imagePullSecrets`	Legt die Abzugsgeheimnisse für das Bild fest `trident-operator`, `trident`, 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: `trace`, `debug`, `info`, `warn`, `error`, Oder `fatal`.	`"info"`
`operatorDebug`	Ermöglicht es, die Protokollebene des Trident-Operators auf Debug zu setzen.	`true`
`operatorImage`	Ermöglicht die vollständige Überschreibung des Bildes für `trident-operator`.	„“
`operatorImageTag`	Ermöglicht das Überschreiben des Tags des `trident-operator` 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`

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.

Wie es weiter geht

Das ist jetzt möglich "Erstellen Sie ein Back-End und eine Storage-Klasse, stellen Sie ein Volume bereit und mounten Sie das Volume in einem POD".

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".

Entscheidende Informationen zu Astra Trident 23.01

Sie müssen die folgenden wichtigen Informationen über Astra Trident lesen.

** Informationen über Astra TripereIT **

- Kubernetes 1.26 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: `23.01.1` Ist die Version des Astra Trident, die Sie installieren.

Bilder in einer Registrierung

```
helm install <name> netapp-trident/trident-operator --version
23.01.1 --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
23.01.1 --set imageRegistry=<your-registry>/sig-storage --set
operatorImage=<your-registry>/netapp/trident-operator:23.01.1 --set
tridentAutosupportImage=<your-registry>/netapp/trident-
autosupport:23.01 --set tridentImage=<your-
registry>/netapp/trident:23.01.1 --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
`--values` (Oder `-f`)	Geben Sie eine YAML-Datei mit Überschreibungen an. Dies kann mehrfach angegeben werden, und die rechteste Datei hat Vorrang.
`--set`	Geben Sie Überschreibungen in der Befehlszeile an.

Um beispielsweise den Standardwert von zu ändern `debug`, Ausführen Sie das folgende `--set` Befehl wo `23.01.1` Ist die Version von Astra Trident, die Sie installieren:

```
helm install <name> netapp-trident/trident-operator --version 23.01.1
--create-namespace --namespace --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
`nodeSelector`	Node-Etiketten für Pod-Zuweisung	
`podAnnotations`	Pod-Anmerkungen	
`deploymentAnnotations`	Anmerkungen zur Bereitstellung	
`tolerations`	Toleranzen für Pod-Zuweisung	
`affinity`	Affinität für Pod-Zuweisung	
`tridentControllerPluginNodeSelector`	Zusätzliche Node-Auswahl für Pods Siehe Allgemeines zu Controller-Pods und Node-Pods Entsprechende Details.	
`tridentControllerPluginTolerations`	Überschreibt Kubernetes-Toleranzen für Pods. Siehe Allgemeines zu Controller-Pods und Node-Pods Entsprechende Details.	
`tridentNodePluginNodeSelector`	Zusätzliche Node-Auswahl für Pods Siehe Allgemeines zu Controller-Pods und Node-Pods Entsprechende Details.	
`tridentNodePluginTolerations`	Überschreibt Kubernetes-Toleranzen für Pods. Siehe Allgemeines zu Controller-Pods und Node-Pods Entsprechende Details.	

Option	Beschreibung	Standard
`imageRegistry`	Identifiziert die Registrierung für den `trident-operator`, `trident`, Und andere Bilder. Lassen Sie das Feld leer, um die Standardeinstellung zu übernehmen.	„“
`imagePullPolicy`	Legt die Richtlinie zum Abziehen von Bildern für den fest `trident-operator`.	`IfNotPresent`
`imagePullSecrets`	Legt die Abzugsgeheimnisse für das Bild fest `trident-operator`, `trident`, 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: `trace`, `debug`, `info`, `warn`, `error`, Oder `fatal`.	`"info"`
`operatorDebug`	Ermöglicht es, die Protokollebene des Trident-Operators auf Debug zu setzen.	`true`
`operatorImage`	Ermöglicht die vollständige Überschreibung des Bildes für `trident-operator`.	„“
`operatorImageTag`	Ermöglicht das Überschreiben des Tags des `trident-operator` 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`

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.

Wie es weiter geht

Das ist jetzt möglich "Erstellen Sie ein Back-End und eine Storage-Klasse, stellen Sie ein Volume bereit und mounten Sie das Volume in einem POD".

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
`namespace`	Namespace für die Installation von Astra Trident in	„Standard“
`debug`	Aktivieren Sie das Debugging für Astra Trident	Falsch
`windows`	Einstellung auf `true` Ermöglicht die Installation auf Windows Worker-Knoten.	Falsch
`IPv6`	Installieren Sie Astra Trident über IPv6	Falsch
`k8sTimeout`	Zeitüberschreitung für Kubernetes-Betrieb	30 Sek.
`silenceAutosupport`	Schicken Sie AutoSupport Bundles nicht automatisch an NetApp	Falsch
`enableNodePrep`	Automatische Verwaltung der Abhängigkeiten von Workers Node (**BETA**)	Falsch
`autosupportImage`	Das Container-Image für AutoSupport Telemetrie	„netapp/Trident-AutoSupport:23.01“
`autosupportProxy`	Die Adresse/der Port eines Proxys zum Senden von AutoSupport Telemetrie	"http://proxy.example.com:8888"
`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:21.04“
`imageRegistry`	Pfad zur internen Registrierung des Formats `<registry FQDN>[:port][/subpath]`	„K8s.gcr.io/sig-Speicherung (k8s 1.19+) oder quay.io/k8scsi“
`kubeletDir`	Pfad zum kubelet-Verzeichnis auf dem Host	„/var/lib/kubelet“
`wipeout`	Eine Liste mit zu löschenden Ressourcen, um Astra Trident vollständig zu entfernen	
`imagePullSecrets`	Secrets, um Bilder aus einer internen Registrierung zu ziehen	

Parameter	Beschreibung	Standard
`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 gleichen Format wie pod.spec.nodeSelector.	Kein Standard; optional
`controllerPluginTolerations`	Überschreibt Kubernetes-Toleranzen für Pods. Entspricht dem gleichen Format wie pod.spec.tolerations.	Kein Standard; optional
`nodePluginNodeSelector`	Zusätzliche Node-Auswahl für Pods Entspricht dem gleichen Format wie pod.spec.nodeSelector.	Kein Standard; optional
`nodePluginTolerations`	Überschreibt Kubernetes-Toleranzen für Pods. Entspricht dem gleichen Format wie pod.spec.tolerations.	Kein Standard; optional

Weitere Informationen zum Formatieren von Pod-Parametern finden Sie unter "Pods werden Nodes zugewiesen".

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
```