Los geht's

Astra Trident

NetApp
April 16, 2024

This PDF was generated from https://docs.netapp.com/de-de/trident-2210/trident-get-started/quickstart.html on April 16, 2024. Always check docs.netapp.com for the latest.

Inhalt

Los geht’s . . . 1
Probieren Sie es aus . . . 1
Anforderungen . . . 1
Implementierungsübersicht . . . 6
Implementierung mit Trident Operator . . . 9
Implementierung mit tridentctl . . . 20
Was kommt als Nächstes? . . . 24

Los geht’s

Probieren Sie es aus

NetApp stellt ein sofort einsatzbereites Lab-Image bereit, das Sie über anfordern können "NetApp Testversion".

Erfahren Sie mehr über die Testversion

Die Testversion bietet eine Sandbox-Umgebung, die mit einem Kubernetes Cluster mit drei Nodes und Astra Trident installiert und konfiguriert ist. Es ist eine gute Möglichkeit, sich mit Astra Trident vertraut zu machen und die Funktionen zu erkunden.

Eine weitere Option ist, die zu sehen "Installationsanleitung für kubeadm" Von Kubernetes bereitgestellt.

> In der Produktion sollte nicht das Kubernetes-Cluster, das Sie erstellen, mit diesen Anweisungen verwendet werden. Verwenden Sie die von der Distribution bereitgestellten Leitfäden zur Implementierung der Produktionsumgebung, um Cluster zu erstellen, die produktionsbereit sind.

Wenn Sie Kubernetes zum ersten Mal verwenden, sollten Sie sich mit den Konzepten und Tools vertraut machen "Hier".

Anforderungen

Prüfen Sie zunächst die unterstützten Frontend-, Back-Ends- und Host-Konfigurationen.

> Weitere Informationen zu den Ports, die Astra Trident verwendet, finden Sie unter "Hier".

Entscheidende Informationen zu Astra Trident 22.10

Vor dem Upgrade auf Astra Trident 22.10 müssen Sie die folgenden wichtigen Informationen lesen.

> **<starke>kritische Informationen über Astra Trident 22.10**
>
> - Kubernetes 1.25 wird jetzt in Trident unterstützt. Vor dem Upgrade auf Kubernetes 1.25 müssen Sie ein Upgrade auf Astra Trident 22.10 durchführen.
> - Astra Trident setzt die Verwendung von Multipathing-Konfiguration in SAN-Umgebungen jetzt strikt um – mit dem empfohlenen Mehrwert 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.

Unterstützte Frontends (Orchestrators)

Astra Trident unterstützt mehrere Container-Engines und Orchestrierungslösungen. Dazu gehören:

- Anthos On-Prem (VMware) und Anthos auf Bare Metal 1.9, 1.10 und 1.11
- Kubernetes 1.20 - 1.25
- Mirantis Kubernetes Engine 3.5
- OpenShift 4.8, 4.9, 4.10, 4.11

Der Trident-Operator wird durch folgende Versionen unterstützt:

- Anthos On-Prem (VMware) und Anthos auf Bare Metal 1.9, 1.10 und 1.11
- Kubernetes 1.20 - 1.25
- OpenShift 4.8, 4.9, 4.10, 4.11

Astra Trident ist auch mit einer Vielzahl weiterer vollständig gemanagter und selbst verwalteter Kubernetes-Angebote kompatibel, darunter Google Kubernetes Engine (GKE), Amazon Elastic Kubernetes Services (EKS), Azure Kubernetes Service (AKS), Rancher und VMware Tanzu Portfolio.

Unterstützte Back-Ends (Storage)

Zur Verwendung von Astra Trident benötigen Sie ein oder mehrere der folgenden unterstützten Back-Ends:

- Amazon FSX für NetApp ONTAP
- Azure NetApp Dateien
- Cloud Volumes ONTAP
- Cloud Volumes Service für GCP
- FAS/All Flash FAS/Select 9.3 oder höher
- NetApp All-SAN-Array (ASA)
- NetApp HCI/Element Software 11 oder höher

Anforderungen an die Funktionen

Die nachfolgende Tabelle enthält einen Überblick über die Funktionen dieser Version von Astra Trident und die von ihm unterstützten Versionen von Kubernetes.

Merkmal	Kubernetes-Version	Funktionstore erforderlich?
CSI Trident	1.20 - 1.25	Nein
Volume Snapshots	1.20 - 1.25	Nein
PVC aus Volume Snapshots	1.20 - 1.25	Nein
ISCSI PV-Größe	1.20 - 1.25	Nein
Bidirektionales ONTAP-CHAP	1.20 - 1.25	Nein
Dynamische Exportrichtlinien	1.20 - 1.25	Nein

Merkmal	Kubernetes-Version	Funktionstore erforderlich?
Trident Operator	1.20 - 1.25	Nein
Auto Worker Node Prep (Beta)	1.20 - 1.25	Nein
CSI-Topologie	1.20 - 1.25	Nein

Getestete Host-Betriebssysteme

Astra Trident unterstützt zwar nicht offiziell bestimmte Betriebssysteme, aber die folgenden sind bekannt:

- Versionen von redhat CoreOS (RHCOS) werden von OpenShift Container Platform unterstützt
- RHEL oder CentOS 7
- Ubuntu 18.04 oder höher (neueste 22.04)
- Windows Server 2019

Standardmäßig wird Astra Trident in einem Container ausgeführt und läuft daher auf jedem Linux-Mitarbeiter. Diese Mitarbeiter müssen jedoch in der Lage sein, die Volumes, die Astra Trident bietet, je nach den von Ihnen verwendeten Back-Ends mit dem standardmäßigen NFS-Client oder iSCSI-Initiator zu mounten.

Der `tridentctl` Utility läuft auch auf jeder dieser Linux-Distributionen.

Host-Konfiguration

Abhängig von den verwendeten Backend(s) sollten NFS und/oder iSCSI Utilities auf allen Arbeitern im Cluster installiert werden. Siehe "Hier" Finden Sie weitere Informationen.

Konfiguration des Storage-Systems

Astra Trident erfordert möglicherweise einige Änderungen an einem Storage-System, bevor eine Backend-Konfiguration verwendet werden kann. Siehe "Hier" Entsprechende Details.

Container-Images und entsprechende Kubernetes-Versionen

Bei luftvergaschten Installationen ist die folgende Liste eine Referenz für Container-Images, die für die Installation von Astra Trident erforderlich sind. Verwenden Sie die `tridentctl images` Befehl zum Überprüfen der Liste der erforderlichen Container-Images.

Kubernetes-Version	Container-Image
V1.20.0	• docker.io/netapp/Trident:22.10.0 • docker.io/netapp/Trident-AutoSupport:22.10 • Registry.k8s.io/SIG-Storage/csi-provisioner:v3.3.0 • Registry.k8s.io/SIG-Storage/csi-Attacher:v4.0.0 • Registry.k8s.io/SIG-Storage/csi-resizer:v1.6.0 • Registry.k8s.io/SIG-Storage/csi-snapshotter:v6.1.0 • Registry.k8s.io/SIG-Storage/csi-Node-driver-Registrar:v2.5.1 • docker.io/netapp/Trident-Operator:22.10.0 (optional)
V1.21,0	• docker.io/netapp/Trident:22.10.0 • docker.io/netapp/Trident-AutoSupport:22.10 • Registry.k8s.io/SIG-Storage/csi-provisioner:v3.3.0 • Registry.k8s.io/SIG-Storage/csi-Attacher:v4.0.0 • Registry.k8s.io/SIG-Storage/csi-resizer:v1.6.0 • Registry.k8s.io/SIG-Storage/csi-snapshotter:v6.1.0 • Registry.k8s.io/SIG-Storage/csi-Node-driver-Registrar:v2.5.1 • docker.io/netapp/Trident-Operator:22.10.0 (optional)
V1.22.0	• docker.io/netapp/Trident:22.10.0 • docker.io/netapp/Trident-AutoSupport:22.10 • Registry.k8s.io/SIG-Storage/csi-provisioner:v3.3.0 • Registry.k8s.io/SIG-Storage/csi-Attacher:v4.0.0 • Registry.k8s.io/SIG-Storage/csi-resizer:v1.6.0 • Registry.k8s.io/SIG-Storage/csi-snapshotter:v6.1.0 • Registry.k8s.io/SIG-Storage/csi-Node-driver-Registrar:v2.5.1 • docker.io/netapp/Trident-Operator:22.10.0 (optional)

Kubernetes-Version	Container-Image
V1.23.0	• docker.io/netapp/Trident:22.10.0 • docker.io/netapp/Trident-AutoSupport:22.10 • Registry.k8s.io/SIG-Storage/csi-provisioner:v3.3.0 • Registry.k8s.io/SIG-Storage/csi-Attacher:v4.0.0 • Registry.k8s.io/SIG-Storage/csi-resizer:v1.6.0 • Registry.k8s.io/SIG-Storage/csi-snapshotter:v6.1.0 • Registry.k8s.io/SIG-Storage/csi-Node-driver-Registrar:v2.5.1 • docker.io/netapp/Trident-Operator:22.10.0 (optional)
V1.24.0	• docker.io/netapp/Trident:22.10.0 • docker.io/netapp/Trident-AutoSupport:22.10 • Registry.k8s.io/SIG-Storage/csi-provisioner:v3.3.0 • Registry.k8s.io/SIG-Storage/csi-Attacher:v4.0.0 • Registry.k8s.io/SIG-Storage/csi-resizer:v1.6.0 • Registry.k8s.io/SIG-Storage/csi-snapshotter:v6.1.0 • Registry.k8s.io/SIG-Storage/csi-Node-driver-Registrar:v2.5.1 • docker.io/netapp/Trident-Operator:22.10.0 (optional)
V1.25.0	• docker.io/netapp/Trident:22.10.0 • docker.io/netapp/Trident-AutoSupport:22.10 • Registry.k8s.io/SIG-Storage/csi-provisioner:v3.3.0 • Registry.k8s.io/SIG-Storage/csi-Attacher:v4.0.0 • Registry.k8s.io/SIG-Storage/csi-resizer:v1.6.0 • Registry.k8s.io/SIG-Storage/csi-snapshotter:v6.1.0 • Registry.k8s.io/SIG-Storage/csi-Node-driver-Registrar:v2.5.1 • docker.io/netapp/Trident-Operator:22.10.0 (optional)

Verwenden Sie in Kubernetes ab Version 1.20 das validierte `registry.k8s.gcr.io/sig-storage/csi-snapshotter:v6.x` Bild nur, wenn der `v1` Version stellt den bereit `volumesnapshots.snapshot.storage.k8s.gcr.io` CRD.- Wenn der `v1beta1` Die Version dient der CRD mit/ohne dem `v1` Verwenden Sie die validierte Version `registry.k8s.gcr.io/sig-storage/csi-snapshotter:v3.x` Bild:

Implementierungsübersicht

Sie können Astra Trident über den Trident-Operator oder mit implementieren `tridentctl`.

Ab Version 22.04 werden die AES-Schlüssel nicht mehr bei jeder Installation von Astra Trident neu generiert. Mit dieser Version installiert Astra Trident ein neues geheimes Objekt, das bei den Installationen fortbesteht. Das bedeutet, `tridentctl` In 22.04 können frühere Versionen von Trident deinstalliert werden, ältere Versionen können jedoch nicht 22.04 Installationen deinstallieren.

Entscheidende Informationen zu Astra Trident 22.10

Vor dem Upgrade auf Astra Trident 22.10 müssen Sie die folgenden wichtigen Informationen lesen.

<starke>kritische Informationen über Astra Trident 22.10

- Kubernetes 1.25 wird jetzt in Trident unterstützt. Vor dem Upgrade auf Kubernetes 1.25 müssen Sie ein Upgrade auf Astra Trident 22.10 durchführen.
- Astra Trident setzt die Verwendung von Multipathing-Konfiguration in SAN-Umgebungen jetzt strikt um – mit dem empfohlenen Mehrwert 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.

Wählen Sie die Bereitstellungsmethode

Um zu ermitteln, welche Bereitstellungsmethode verwendet werden soll, sollten Sie Folgendes berücksichtigen:

Wann der Trident-Operator verwendet werden soll

Der "Betreiber von Trident" Bietet eine hervorragende Möglichkeit, Astra Trident Ressourcen dynamisch zu managen und die Einrichtungsphase zu automatisieren. Es gibt einige Voraussetzungen, die erfüllt werden müssen. Siehe "Den Anforderungen gerecht zu werden".

Der Trident Operator hat mehrere Vorteile wie unten beschrieben.

Funktionen zur Selbstreparatur

Sie können eine Astra Trident-Installation überwachen und aktiv Maßnahmen ergreifen, um Probleme wie das

Löschen der Implementierung oder das versehentliche Ändern der Implementierung zu beheben. Wenn der Bediener als Bereitstellung eingerichtet ist, wird ein `trident-operator-<generated-id>` Pod wird erstellt. Dieser Pod ordnet A zu `TridentOrchestrator` CR mit einer Astra Trident Installation und sorgt stets dafür, dass nur eine aktiv ist `TridentOrchestrator`. Mit anderen Worten, der Operator stellt sicher, dass es nur eine Instanz von Astra Trident im Cluster gibt und steuert seine Einrichtung, um sicherzustellen, dass die Installation ist idempotent. Wenn Änderungen an der Installation vorgenommen werden (z. B. Löschen der Bereitstellung oder Knotendemonsatz), identifiziert der Bediener diese und korrigiert sie einzeln.

Einfache Updates vorhandener Installationen

Sie können eine vorhandene Implementierung einfach mit dem Bediener aktualisieren. Sie müssen nur die bearbeiten `TridentOrchestrator` CR, um Aktualisierungen für eine Installation durchzuführen. Betrachten Sie zum Beispiel ein Szenario, bei dem Sie Astra Trident aktivieren müssen, um Debug-Protokolle zu generieren.

Um dies zu tun, patchen Sie Ihre `TridentOrchestrator` Einstellen `spec.debug` Bis `true`:

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

Nachher `TridentOrchestrator` Wird aktualisiert, verarbeitet der Bediener die Updates und Patches für die vorhandene Installation. Dies kann dazu führen, dass neue Pods erstellt werden, um die Installation entsprechend zu ändern.

Automatische Verarbeitung von Kubernetes-Upgrades

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

> Wenn das Cluster auf eine nicht unterstützte Version aktualisiert wird, verhindert der Operator die Installation von Astra Trident. Falls Astra Trident bereits mit dem Operator installiert wurde, wird eine Warnmeldung angezeigt, die angibt, dass Astra Trident auf einer nicht unterstützten Kubernetes-Version installiert ist.

Verwalten Sie Ihre Kubernetes-Cluster mit BlueXP (ehemals Cloud Manager).

Mit "Astra Trident mit BlueXP", Sie können ein Upgrade auf die neueste Version von Astra Trident durchführen, Storage-Klassen hinzufügen und managen, mit Arbeitsumgebungen verbinden und persistente Volumes mit Cloud Backup Service sichern. BlueXP unterstützt die Astra Trident-Implementierung mithilfe des Trident-Operators entweder manuell oder über Helm.

Wann soll der Helm verwendet werden

Wenn Sie andere Applikationen, die Sie mit Helm managen, ab Astra Trident 21.01 können Sie Ihre Implementierung auch mit Helm managen.

Wann verwendet werden soll `tridenctl`

Wenn Sie eine vorhandene Implementierung haben, die aktualisiert werden muss oder Sie Ihre Implementierung stark anpassen möchten, sollten Sie den Einsatz in Betracht ziehen "Tridentctl". Dies ist die herkömmliche Methode der Implementierung von Astra Trident.

Überlegungen beim Wechsel zwischen Implementierungsmethoden

Es ist nicht schwer, sich ein Szenario vorzustellen, in dem ein Wechsel zwischen Implementierungsmethoden gewünscht wird. Sie sollten Folgendes berücksichtigen, bevor Sie versuchen, von A zu wechseln `tridentctl` Implementierung in eine Operator-basierte Implementierung oder umgekehrt:

- Verwenden Sie immer die gleiche Methode, um Astra Trident zu deinstallieren. Wenn Sie mit bereitgestellt haben `tridentctl`, Sie sollten die entsprechende Version des verwenden `tridentctl` Binary zur Deinstallation von Astra Trident. Ebenso sollten Sie bei der Bereitstellung mit dem Operator die bearbeiten `TridentOrchestrator` CR und Set `spec.uninstall=true` Um Astra Trident zu deinstallieren.
- Wenn Sie über eine bedienerbasierte Bereitstellung verfügen, die Sie entfernen und verwenden möchten `tridentctl` Bei der Implementierung von Astra Trident sollten Sie zuerst bearbeiten `TridentOrchestrator` Und gesetzt `spec.uninstall=true` Um Astra Trident zu deinstallieren. Löschen Sie dann `TridentOrchestrator` Und die Bedienerbereitstellung. Sie können dann mit installieren `tridentctl`.
- Wenn Sie über eine manuelle, bedienerbasierte Implementierung verfügen und die Helm-basierte Trident Operator-Implementierung verwenden möchten, sollten Sie zuerst den Operator manuell deinstallieren und dann die Helm-Installation durchführen. So kann Helm den Trident-Operator mit den erforderlichen Beschriftungen und Anmerkungen implementieren. Wenn dies nicht der Fall ist, schlägt die Bereitstellung des Helm-basierten Trident-Operators mit einem Fehler bei der Labelvalidierung und einem Validierungsfehler bei der Annotation fehl. Wenn Sie eine haben `tridentctl`-Basierte Bereitstellung, können Sie Helm-basierte Implementierung nutzen, ohne Probleme zu verursachen.

Analysieren Sie die Bereitstellungsmodi

Es gibt drei Möglichkeiten für die Implementierung von Astra Trident:

Standardimplementierung

Die Implementierung von Trident auf einem Kubernetes Cluster führt zum Astra Trident Installer auf zwei Dinge:

- Abrufen der Container-Images über das Internet
- Erstellung einer Implementierung und/oder Node-Demonset, bei der Astra Trident Pods auf allen teilnahmeberechtigten Nodes im Kubernetes-Cluster gespinnt werden.

Eine solche Standardimplementierung kann auf zwei verschiedene Arten ausgeführt werden:

- Wird Verwendet `tridentctl install`
- Verwenden des Betreibers von Trident. Trident-Operator kann entweder manuell oder mit Helm implementiert werden.

Dieser Installationsmodus ist die einfachste Möglichkeit, Astra Trident zu installieren und funktioniert für die meisten Umgebungen, die keine Netzwerkeinschränkungen auferlegen.

Offline-Bereitstellung

Um eine luftvergoppte Installation durchzuführen, können Sie den verwenden `--image-registry` Markierung beim Aufrufen `tridentctl install` Auf eine private Bildregistrierung verweisen. Bei der Implementierung mit dem Trident-Operator können Sie alternativ angeben `spec.imageRegistry` In Ihren `TridentOrchestrator`. Diese Registrierung sollte den enthalten "Bild: Trident", Das "Bild: Trident AutoSupport", Und die CSI-Sidecar-Bilder, wie von Ihrer Kubernetes-Version erforderlich.

Verwenden Sie zum Anpassen Ihrer Implementierung die Möglichkeit `tridentctl` Generierung der Manifeste für Trident Ressourcen: Dies umfasst die Implementierung, das Demonet, das Servicekonto und die Cluster-Rolle, die Astra Trident im Rahmen der Installation erstellt.

Weitere Informationen zum Anpassen Ihrer Bereitstellung finden Sie unter diesen Links:

- "Anpassung der benutzerbasierten Implementierung"
  *

> Wenn Sie ein privates Image Repository verwenden, sollten Sie hinzufügen `/sig-storage` Bis zum Ende der privaten Registrierungs-URL. Wenn Sie eine private Registrierung für verwenden `tridentctl` Implementierung, sollten Sie verwenden `--trident-image` Und `--autosupport-image` Zusammen mit `--image-registry`. Wenn Sie Astra Trident mithilfe des Trident-Operators implementieren, stellen Sie sicher, dass der Orchestrator CR enthält `tridentImage` Und `autosupportImage` In den Installationsparametern.

Remote-Implementierung

Im Folgenden finden Sie einen allgemeinen Überblick über den Remote-Implementierungsprozess:

- Stellen Sie die entsprechende Version von bereit `kubectl` Auf dem Remote-Rechner, von wo aus Sie Astra Trident implementieren möchten.
- Kopieren Sie die Konfigurationsdateien aus dem Kubernetes-Cluster und legen Sie die fest `KUBECONFIG` Umgebungsvariable auf dem Remotecomputer.
- Initiieren Sie A `kubectl get nodes` Befehl zum Überprüfen, ob eine Verbindung mit dem erforderlichen Kubernetes-Cluster hergestellt werden kann.
- Führen Sie die Implementierung von der Remote-Maschine aus, indem Sie die standardmäßigen Installationsschritte verwenden.

Andere bekannte Konfigurationsoptionen

Bei der Installation von Astra Trident auf VMware Tanzu Portfolio Produkten:

- Das Cluster muss privilegierte Workloads unterstützen.
- Der `--kubelet-dir` Flag sollte auf den Speicherort des kubelet-Verzeichnisses gesetzt werden. Standardmäßig ist dies `/var/vcap/data/kubelet`.

  Festlegen der Kubelet-Position unter Verwendung `--kubelet-dir` Ist für Trident Operator, Helm und bekannt `tridentctl` Implementierungen.

Implementierung mit Trident Operator

Sie können Astra Trident über den Trident-Operator implementieren.

Entscheidende Informationen zu Astra Trident 22.10

Vor dem Upgrade auf Astra Trident 22.10 müssen Sie die folgenden wichtigen Informationen lesen.

<starke>kritische Informationen über Astra Trident 22.10

- Kubernetes 1.25 wird jetzt in Trident unterstützt. Vor dem Upgrade auf Kubernetes 1.25 müssen Sie ein Upgrade auf Astra Trident 22.10 durchführen.
- Astra Trident setzt die Verwendung von Multipathing-Konfiguration in SAN-Umgebungen jetzt strikt um – mit dem empfohlenen Mehrwert 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.

Implementierungsoptionen für Trident Operator

Es gibt zwei Möglichkeiten zur Implementierung des Trident Operators:

- Verwendung von Trident "Steuerruderdiagramm": Das Helm Chart implementiert den Trident-Operator und installiert Trident in einem Schritt.
- Manuell: Trident stellt eine Datei zur Verfügung, mit der der Operator installiert und zugehörige Objekte erstellt werden können.
  - 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".

Wenn Sie sich nicht bereits mit dem vertraut gemacht haben "Grundkonzepte", Ist jetzt eine tolle Zeit, um das zu tun.

Voraussetzungen prüfen

Bei der Implementierung von Astra Trident sollten die folgenden Voraussetzungen erfüllt sein:

- Sie erhalten vollständige Berechtigungen für einen unterstützten Kubernetes-Cluster mit einer unterstützten Version von Kubernetes. Überprüfen Sie die "Anforderungen".
- Sie haben Zugriff auf ein unterstütztes NetApp Storage-System.
- Sie können Volumes von allen Kubernetes-Worker-Nodes einbinden.
- Sie verfügen über einen Linux-Host mit `kubectl` (Oder `oc`, Falls Sie OpenShift nutzen) ist installiert und konfiguriert, um den Kubernetes-Cluster zu managen, den Sie verwenden möchten.
- Sie haben die festgelegt `KUBECONFIG` Umgebungsvariable auf die Kubernetes-Cluster-Konfiguration verweisen.
- Sie haben das aktiviert "Funktionsgates erforderlich von Astra Trident".
- Bei Verwendung von Kubernetes mit Docker Enterprise "Führen Sie die entsprechenden Schritte aus, um den CLI-Zugriff zu aktivieren".

Hast du das alles? Sehr Gut! Fangen wir an.

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

Führen Sie die aufgeführten Schritte zur Implementierung des Trident-Bedieners mithilfe von Helm durch.

Was Sie benötigen

Zusätzlich zu den oben aufgeführten Voraussetzungen benötigen Sie zur Implementierung des Trident-Operators mithilfe von Helm folgende Voraussetzungen:

- A "Unterstützte Kubernetes-Version"
- Helm Version 3

Schritte

1. Helm-Repository von Trident hinzufügen:

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

2. Verwenden Sie die `helm install` Führen Sie einen Befehl aus, und geben Sie einen Namen für Ihre Bereitstellung an. Das folgende Beispiel zeigt:

```
helm install <name> netapp-trident/trident-operator --version 22.10.0
--create-namespace --namespace <trident-namespace>
```

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

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

- `--values` (Oder `-f`): Geben Sie eine YAML-Datei mit Überschreibungen an. Dies kann mehrfach angegeben werden, und die rechteste Datei hat Vorrang.
- `--set`: Überschreibungen auf der Kommandozeile angeben.

Um beispielsweise den Standardwert von zu ändern `debug`, Ausführen Sie das folgende `--set` Befehl:

```
helm install <name> netapp-trident/trident-operator --version 22.10.0
--create-namespace --namespace --set tridentDebug=true
```

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

`helm list` Zeigt Ihnen Details zur Installation an, z. B. Name, Namespace, Diagramm, Status, App-Version, Revisionsnummer usw.

Trident Operator kann manuell implementiert und Trident installiert werden

Führen Sie die aufgeführten Schritte aus, um den Trident-Operator manuell zu implementieren.

Schritt: Qualifizieren Sie Ihren Kubernetes-Cluster

Zunächst müssen Sie sich beim Linux-Host anmelden und überprüfen, ob es ein *Working*, "Unterstützter Kubernetes-Cluster" Dass Sie über die erforderlichen Berechtigungen verfügen.

> 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`.

Führen Sie den folgenden Befehl aus, um Ihre Kubernetes-Version zu überprüfen:

```
kubectl version
```

So zeigen Sie, ob Sie über die Berechtigungen für Kubernetes Cluster-Administratoren verfügen:

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

Führen Sie den folgenden Befehl aus, um zu überprüfen, ob ein POD mit einem Image aus dem Docker Hub gestartet werden kann und das Storage-System über das POD-Netzwerk erreichen kann:

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

Schritt 2: Laden Sie den Operator herunter und richten Sie ihn ein

> Ab 21.01 ist der Trident Operator der Cluster-Umfang. Zur Installation von Trident muss mit dem Trident-Operator der erstellt werden `TridentOrchestrator` Benutzerdefinierte Ressourcendefinition (CRD) und Definition anderer Ressourcen. Führen Sie diese Schritte durch, um den Bediener einzurichten, bevor Sie Astra Trident installieren können.

1. Laden Sie die neueste Version des Trident Installationspakets herunter und extrahieren Sie "Die Sektion *Assets* auf GitHub".

   ```
   wget
   https://github.com/NetApp/trident/releases/download/v22.10.0/trident-
   installer-22.10.0.tar.gz
   tar -xf trident-installer-22.10.0.tar.gz
   cd trident-installer
   ```

2. Verwenden Sie das entsprechende CRD-Manifest, um das zu erstellen `TridentOrchestrator` CRD.- Dann erstellen Sie ein `TridentOrchestrator` Benutzerdefinierte Ressource später, um eine Installation durch den Bediener zu erstellen.

   Führen Sie den folgenden Befehl aus:

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

3. Nach dem `TridentOrchestrator` CRD wird erstellt, so legen Sie die folgenden Ressourcen für die Bedienerbereitstellung an:
   - Ein Servicekonto für den Betreiber
   - ClusterRPole und ClusterRoleBending an das ServiceAccount
   - Eine dedizierte PodSecurityPolicy
   - Der Bediener selbst

     Trident Installer enthält Manifeste für die Definition dieser Ressourcen. Standardmäßig wird der Operator in bereitgestellt `trident` Namespace. Wenn der `trident` Der Namespace ist nicht vorhanden. Verwenden Sie das folgende Manifest, um einen zu erstellen.

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

4. So stellen Sie den Operator in einem anderen Namespace als dem Standard bereit `trident` Namespace, sollten Sie den aktualisieren `serviceaccount.yaml`, `clusterrolebinding.yaml` Und `operator.yaml` Manifeste und Generate Your `bundle.yaml`.

   Führen Sie den folgenden Befehl aus, um die YAML Manifeste zu aktualisieren und das zu generieren `bundle.yaml` Verwenden der `kustomization.yaml`:

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

   Führen Sie den folgenden Befehl aus, um die Ressourcen zu erstellen und den Operator bereitzustellen:

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

5. Gehen Sie wie folgt vor, um den Status des Bedieners nach der Bereitstellung zu überprüfen:

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

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

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

NAME                                READY   STATUS              RESTARTS
AGE
trident-operator-54cb664d-lnjxh     1/1     Running             0
3m
```

Durch die Implementierung eines Mitarbeiters wird erfolgreich ein Pod erstellt, der auf einem der Worker-

Nodes im Cluster ausgeführt wird.

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

Schritt 3: Erstellen `TridentOrchestrator` Und Trident installieren

Sie können Astra Trident nun mit dem Operator installieren! Hierfür muss erstellt werden `TridentOrchestrator`. Das Trident Installationsprogramm enthält Beispieldefinitionen für die Erstellung `TridentOrchestrator`. Dies startet eine Installation im `trident` Namespace.

```
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:22.10
    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:21.04.0
  Message:                  Trident installed  Namespace:
trident
  Status:                   Installed
  Version:                  v21.04.0
Events:
    Type Reason Age From Message ---- ------ ---- ---- -------Normal
    Installing 74s trident-operator.netapp.io Installing Trident Normal
    Installed 67s trident-operator.netapp.io Trident installed
```

Der Trident-Operator ermöglicht es Ihnen, die Art und Weise, wie Astra Trident installiert wird, mithilfe der Attribute im anzupassen `TridentOrchestrator` Spez. Siehe "Anpassung der Trident Implementierung".

Der Status von `TridentOrchestrator` Gibt an, ob die Installation erfolgreich war und zeigt die installierte Version von Trident an.

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.

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. Sie sollten die Protokolle des Operators überprüfen. Siehe "Fehlerbehebung" Abschnitt.

Sie können überprüfen, ob die Astra Trident Installation abgeschlossen wurde, indem Sie sich die erstellten Pods ansehen:

```
kubectl get pod -n trident

NAME                                READY   STATUS    RESTARTS   AGE
trident-csi-7d466bf5c7-v4cpw        5/5     Running   0           1m
trident-csi-mr6zc                   2/2     Running   0           1m
trident-csi-xrp7w                   2/2     Running   0           1m
trident-csi-zh2jt                   2/2     Running   0           1m
trident-operator-766f7b8658-ldzsv   1/1     Running   0           3m
```

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

```
./tridentctl -n trident version

+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 21.04.0        | 21.04.0        |
+----------------+----------------+
```

Jetzt können Sie mit diesem Schritt ein Backend erstellen. Siehe "Aufgaben nach der Implementierung".

Informationen zur Fehlerbehebung bei Problemen während der Bereitstellung finden Sie im "Fehlerbehebung" Abschnitt.

Anpassung der Trident Operator-Implementierung

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, sollten Sie erwägen, zu verwenden `tridentctl` So generieren Sie benutzerdefinierte YAML-Manifeste, die Sie nach Bedarf ändern können.

`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.

Konfigurationsoptionen

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:22.10.0“
`autosupportProxy`	Die Adresse/der Port eines Proxys zum Senden von AutoSupport Telemetrie	"http://proxy.example.com:8888"

Parameter	Beschreibung	Standard
`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	
`controllerPluginNodeSelector`	Zusätzliche Node-Selektoren für Pods mit dem Trident Controller CSI Plugin Entspricht dem gleichen Format wie pod.spec.nodeSelector.	Kein Standard; optional
`controllerPluginTolerations`	Überschreibungen von Verträgungen für Pods mit dem Trident Controller CSI-Plug-in Entspricht dem gleichen Format wie pod.spec.tolerations.	Kein Standard; optional
`nodePluginNodeSelector`	Zusätzliche Node-Selektoren für Pods, auf denen das Trident Node CSI Plugin ausgeführt wird. Entspricht dem gleichen Format wie pod.spec.nodeSelector.	Kein Standard; optional
`nodePluginTolerations`	Überschreibungen von Verträgungen für Pods mit dem Trident Node CSI Plugin 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
```

Implementierung mit tridentctl

Astra Trident ist mit verfügbar `tridentctl`. Es ist eine gute Idee, sich mit dem vertraut zu machen "Grundkonzepte". Um Ihre anzupassen `tridentctl` Die Bereitstellung finden Sie unter "Tridentctl-Implementierung anpassen".

Entscheidende Informationen zu Astra Trident 22.10

Vor dem Upgrade auf Astra Trident 22.10 müssen Sie die folgenden wichtigen Informationen lesen.

<starke>kritische Informationen über Astra Trident 22.10

- Kubernetes 1.25 wird jetzt in Trident unterstützt. Vor dem Upgrade auf Kubernetes 1.25 müssen Sie ein Upgrade auf Astra Trident 22.10 durchführen.
- Astra Trident setzt die Verwendung von Multipathing-Konfiguration in SAN-Umgebungen jetzt strikt um – mit dem empfohlenen Mehrwert 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.

Voraussetzungen prüfen

Bei der Implementierung von Astra Trident sollten die folgenden Voraussetzungen erfüllt sein:

- Volle Berechtigungen für ein unterstütztes Kubernetes-Cluster.
- Zugriff auf ein unterstütztes NetApp Storage-System.
- Kann Volumes von allen Kubernetes Worker-Nodes aus mounten
- Einem Linux-Host mit `kubectl` (Oder `oc`, Falls Sie OpenShift nutzen) ist installiert und konfiguriert, um den Kubernetes-Cluster zu managen, den Sie verwenden möchten.

- Der `KUBECONFIG` Umgebungsvariable verweist auf Ihre Kubernetes-Cluster-Konfiguration.
- Der "Funktionsgates erforderlich von Astra Trident" Aktiviert sind.
- Bei Verwendung von Kubernetes mit Docker Enterprise "Führen Sie die entsprechenden Schritte aus, um den CLI-Zugriff zu aktivieren".

Schritt: Qualifizieren Sie Ihren Kubernetes-Cluster

Melden Sie sich beim Linux-Host an und überprüfen Sie, ob er einen funktionierenden, "Unterstützter Kubernetes-Cluster" Und Sie haben die erforderlichen Berechtigungen.

> Mit OpenShift ist Ihr Einsatz `oc` Statt `kubectl` In allen folgenden Beispielen sollten Sie sich zuerst als **System:admin** anmelden, indem Sie ausführen `oc login -u system:admin` Oder `oc login -u kube-admin`.

So prüfen Sie Ihre Kubernetes-Version:

```
kubectl version
```

Führen Sie den folgenden Befehl aus, um die Berechtigungen des Kubernetes-Cluster-Administrators zu überprüfen:

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

Führen Sie den folgenden Befehl aus, um zu überprüfen, ob ein POD mit einem Image aus dem Docker Hub gestartet werden kann und das Storage-System über das POD-Netzwerk erreichen kann:

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

Ermitteln Sie die Kubernetes-Serverversion. Sie verwenden es bei der Installation von Astra Trident.

Schritt 2: Downloaden und extrahieren Sie das Installationsprogramm

> Das Trident-Installationsprogramm erstellt ein Trident Pod, konfiguriert die CRD-Objekte, die zum Erhalt seines Status verwendet werden, und initialisiert die CSI-Sidecars, die Aktionen ausführen, wie z. B. die Bereitstellung und das Anschließen von Volumes an Cluster-Hosts.

Sie können die neueste Version des Trident Installationspakets herunterladen und extrahieren "Die Sektion *Assets* auf GitHub".

Beispiel: Wenn die neueste Version 22.10.0 ist:

```
wget https://github.com/NetApp/trident/releases/download/v22.10.0/trident-
installer-22.10.0.tar.gz
tar -xf trident-installer-22.10.0.tar.gz
cd trident-installer
```

Schritt 3: Installieren Sie Astra Trident

Installieren Sie Astra Trident im gewünschten Namespace, indem Sie den ausführen `tridentctl install` Befehl.

```
./tridentctl install -n trident
....
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-csi-679648bd45-cv2mx
INFO Waiting for Trident REST interface.
INFO Trident REST interface is up.                 version=22.10.0
INFO Trident installation succeeded.
....
```

> Damit Astra Trident auf Windows-Knoten ausgeführt werden kann, fügen Sie die hinzu `--windows` Flag auf den Installationsbefehl: `$ ./tridentctl install --windows -n trident`.

Die Ausgabe wird ähnlich wie die folgenden angezeigt, wenn das Installationsprogramm abgeschlossen ist. Abhängig von der Anzahl der Nodes in Ihrem Kubernetes Cluster können hier mehr Pods vorhanden sein:

```
kubectl get pod -n trident
NAME                              READY   STATUS    RESTARTS   AGE
trident-csi-679648bd45-cv2mx      4/4     Running   0          5m29s
trident-csi-vgc8n                 2/2     Running   0          5m29s

./tridentctl -n trident version
+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 22.10.0        | 22.10.0        |
+----------------+----------------+
```

Um die Konfiguration von Astra Trident abzuschließen, fahren Sie fort "Aufgaben nach der Implementierung".

Wenn das Installationsprogramm nicht erfolgreich abgeschlossen wurde, oder `trident-csi-<generated id>` Hat keinen **running** Status, die Plattform wurde nicht installiert.

> Informationen zur Fehlerbehebung bei Problemen während der Implementierung finden Sie unter "Fehlerbehebung".

Tridentctl-Implementierung anpassen

Mit dem Astra Trident Installer können Sie die Implementierung anpassen.

Erfahren Sie mehr über das Installationsprogramm

Mit dem Astra Trident Installer können Sie Attribute anpassen. Wenn Sie beispielsweise das Trident-Image in ein privates Repository kopiert haben, können Sie den Bildnamen mithilfe von angeben `--trident-image`. Wenn Sie das Trident-Image sowie die erforderlichen CSI-Sidecar-Images in ein privates Repository kopiert haben, ist es möglicherweise besser, den Speicherort des Repository mithilfe von anzugeben `--image-registry` Schalter, der die Form nimmt `<registry FQDN>[:port]`.

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

Wenn Sie die Installation anpassen müssen, die über die Argumente des Installers hinausgeht, können Sie auch die Bereitstellungsdateien anpassen. Verwenden der `--generate-custom-yaml` Der Parameter erstellt die folgenden YAML-Dateien im Installationsprogramm `setup` Verzeichnis:

- `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 erstellt haben, können Sie sie nach Ihren Bedürfnissen ändern und dann verwenden `--use-custom-yaml` Um Ihre benutzerdefinierte Bereitstellung zu installieren.

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

Was kommt als Nächstes?

Nach der Implementierung von Astra Trident können Sie mit der Erstellung eines Backend, der Erstellung einer Storage-Klasse, der Bereitstellung eines Volumes und dem Mounten des Volumes in einem Pod fortfahren.

Schritt 1: Erstellen Sie ein Backend

Jetzt können Sie mit Astra Trident ein Backend erstellen und Volumes bereitstellen. Erstellen Sie dazu ein `backend.json` Datei, die die erforderlichen Parameter enthält. Beispiele für Konfigurationsdateien für verschiedene Backend-Typen finden Sie im `sample-input` Verzeichnis.

Siehe "Hier" Weitere Informationen zum Konfigurieren der Datei für den Backend-Typ.

```
cp sample-input/<backend template>.json backend.json
vi backend.json
```

```
./tridentctl -n trident create backend -f backend.json
+-------------+----------------+--------------------------------------
+--------+---------+
|    NAME     | STORAGE DRIVER |                 UUID                 |
STATE  | VOLUMES |
+-------------+----------------+--------------------------------------
+--------+---------+
| nas-backend | ontap-nas      | 98e19b74-aec7-4a3d-8dcf-128e5033b214 |
online |       0 |
+-------------+----------------+--------------------------------------
+--------+---------+
```

Wenn die Erstellung fehlschlägt, ist mit der Back-End-Konfiguration ein Fehler aufgetreten. Sie können die Protokolle zur Bestimmung der Ursache anzeigen, indem Sie den folgenden Befehl ausführen:

```
./tridentctl -n trident logs
```

Nachdem Sie das Problem behoben haben, gehen Sie einfach zurück zum Anfang dieses Schritts und versuchen Sie es erneut. Weitere Tipps zur Fehlerbehebung finden Sie unter "Die Fehlerbehebung" Abschnitt.

Schritt 2: Erstellen Sie eine Storage-Klasse

Kubernetes Benutzer stellen Volumes mithilfe von persistenten Volume Claims (PVCs) bereit, die einen angeben "Storage-Klasse" Nach Name. Die Details sind für die Benutzer verborgen. In einer Storage-Klasse wird jedoch die provisionierung für die jeweilige Klasse (in diesem Fall Trident) angegeben, und die Bedeutung dieser Klasse für die provisionierung angegeben.

Kubernetes-Benutzer in Storage-Klasse erstellen geben an, wann sie ein Volume möchten. Die Konfiguration der Klasse muss das im vorherigen Schritt erstellte Backend modellieren, damit Astra Trident neue Volumes bereitstellen wird.

Die einfachste Storage-Klasse, mit der Sie beginnen können, basiert auf der `sample-input/storage-class-csi.yaml.templ` Datei, die mit dem Installationsprogramm geliefert wird, ersetzen *`BACKEND_TYPE`* Mit dem Namen des Speichertreibers.

```
./tridentctl -n trident get backend
+-------------+----------------+--------------------------------------
+--------+---------+
|    NAME     | STORAGE DRIVER |                 UUID                 |
STATE   | VOLUMES |
+-------------+----------------+--------------------------------------
+--------+---------+
| nas-backend | ontap-nas      | 98e19b74-aec7-4a3d-8dcf-128e5033b214 |
online |       0 |
+-------------+----------------+--------------------------------------
+--------+---------+

cp sample-input/storage-class-csi.yaml.templ sample-input/storage-class-
basic-csi.yaml

# Modify __BACKEND_TYPE__ with the storage driver field above (e.g.,
ontap-nas)
vi sample-input/storage-class-basic-csi.yaml
```

Dies ist ein Kubernetes-Objekt, die Storage-Verwendung ist `kubectl` Um sie in Kubernetes zu erstellen.

```
kubectl create -f sample-input/storage-class-basic-csi.yaml
```

Sie sollten jetzt in Kubernetes und Astra Trident eine **Basis-csi** Storage-Klasse sehen, und Astra Trident hätte die Pools auf dem Backend entdeckt haben sollen.

```
kubectl get sc basic-csi
NAME         PROVISIONER             AGE
basic-csi    csi.trident.netapp.io   15h

./tridentctl -n trident get storageclass basic-csi -o json
{
  "items": [
    {
      "Config": {
        "version": "1",
        "name": "basic-csi",
        "attributes": {
          "backendType": "ontap-nas"
        },
        "storagePools": null,
        "additionalStoragePools": null
      },
      "storage": {
        "ontapnas_10.0.0.1": [
          "aggr1",
          "aggr2",
          "aggr3",
          "aggr4"
        ]
      }
    }
  ]
}
```

Schritt 3: Stellen Sie Ihr erstes Volumen bereit

Nun können Sie das erste Volume dynamisch bereitstellen. Dazu wird ein Kubernetes erstellt "Persistent Volume Claim" (PVC) Objekt.

Erstellen Sie eine PVC für ein Volume, das die soeben erstellte Storage-Klasse verwendet.

Siehe `sample-input/pvc-basic-csi.yaml` Beispiel: Stellen Sie sicher, dass der Name der Speicherklasse mit dem übereinstimmt, den Sie erstellt haben.

```
kubectl create -f sample-input/pvc-basic-csi.yaml

kubectl get pvc --watch
NAME      STATUS    VOLUME                                     CAPACITY
ACCESS MODES  STORAGECLASS   AGE
basic     Pending
basic          1s
basic     Pending   pvc-3acb0d1c-b1ae-11e9-8d9f-5254004dfdb7   0
basic          5s
basic     Bound     pvc-3acb0d1c-b1ae-11e9-8d9f-5254004dfdb7   1Gi
RWO           basic          7s
```

Schritt 4: Mounten Sie die Volumes in einem POD

Nun lassen Sie uns den Datenträger einhängen. Wir werden einen nginx-Pod starten, der das PV unter einhängt `/usr/share/nginx/html`.

```
cat << EOF > task-pv-pod.yaml
kind: Pod
apiVersion: v1
metadata:
  name: task-pv-pod
spec:
  volumes:
    - name: task-pv-storage
      persistentVolumeClaim:
       claimName: basic
  containers:
    - name: task-pv-container
      image: nginx
      ports:
        - containerPort: 80
          name: "http-server"
      volumeMounts:
        - mountPath: "/usr/share/nginx/html"
          name: task-pv-storage
EOF
kubectl create -f task-pv-pod.yaml
```

```
# Wait for the pod to start
kubectl get pod --watch

# Verify that the volume is mounted on /usr/share/nginx/html
kubectl exec -it task-pv-pod -- df -h /usr/share/nginx/html

# Delete the pod
kubectl delete pod task-pv-pod
```

An diesem Punkt existiert der POD (Applikation) nicht mehr, das Volume ist jedoch weiterhin vorhanden. Sie können es von einem anderen POD nutzen, wenn Sie dies möchten.

Löschen Sie zum Löschen des Volumes die Forderung:

```
kubectl delete pvc basic
```

Sie können jetzt zusätzliche Aufgaben ausführen, wie z. B.:

- "Konfigurieren Sie zusätzliche Back-Ends."
- "Erstellen Sie zusätzliche Speicherklassen."

Copyright-Informationen

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