



Astra Trident 23.07 Dokumentation

Astra Trident

NetApp
April 03, 2024

Inhalt

| | |
|---|-----|
| Astra Trident 23.07 Dokumentation | 1 |
| Versionshinweise | 2 |
| Was ist neu | 2 |
| Frühere Versionen der Dokumentation | 12 |
| Los geht's | 13 |
| Erfahren Sie mehr über Astra Trident | 13 |
| Der Einstieg in Astra Trident ist schnell möglich | 19 |
| Anforderungen | 21 |
| Installation Von Astra Trident | 26 |
| Erfahren Sie mehr über die Installation von Astra Trident | 26 |
| Installation über den Trident Operator | 30 |
| Installieren Sie mit tridentctl | 56 |
| Nutzen Sie Astra Trident | 61 |
| Bereiten Sie den Knoten „Worker“ vor | 61 |
| Konfiguration und Management von Back-Ends | 65 |
| Erstellen und Managen von Storage-Klassen | 181 |
| Provisionierung und Management von Volumes | 186 |
| Management und Monitoring von Astra Trident | 223 |
| Upgrade Astra Trident | 223 |
| Überwachen Sie Astra Trident | 229 |
| Deinstallieren Sie Astra Trident | 233 |
| Astra Trident für Docker | 236 |
| Voraussetzungen für die Bereitstellung | 236 |
| Implementieren Sie Astra Trident | 239 |
| Astra Trident upgraden oder deinstallieren | 243 |
| Arbeiten mit Volumes | 245 |
| Sammelt Protokolle | 254 |
| Management mehrerer Astra Trident Instanzen | 255 |
| Optionen für die Storage-Konfiguration | 256 |
| Bekannte Probleme und Einschränkungen | 265 |
| Best Practices und Empfehlungen | 267 |
| Einsatz | 267 |
| Storage-Konfiguration | 267 |
| Integration Von Astra Trident | 274 |
| Datensicherung und Disaster Recovery | 285 |
| Sicherheit | 288 |
| Wissen und Support | 295 |
| Häufig gestellte Fragen | 295 |
| Fehlerbehebung | 302 |
| Unterstützung | 307 |
| Referenz | 310 |
| Astra Trident-Ports | 310 |
| Astra Trident REST-API | 310 |

| | |
|---|-----|
| Befehlszeilenoptionen | 311 |
| NetApp Produkte sind in Kubernetes integriert | 312 |
| Kubernetes und Trident Objekte | 313 |
| Tridentctl-Befehle und -Optionen | 326 |
| Pod Security Standards (PSS) und Security Context Constraints (SCC) | 331 |
| Rechtliche Hinweise | 337 |
| Urheberrecht | 337 |
| Marken | 337 |
| Patente | 337 |
| Datenschutzrichtlinie | 337 |
| Open Source | 337 |

Astra Trident 23.07 Dokumentation

Versionshinweise

Was ist neu

Versionshinweise liefern Informationen zu den neuen Funktionen, Verbesserungen und Bugfixes in der aktuellen Version von Astra Trident.



Der `tridentctl` Binary for Linux, die in der ZIP-Datei des Installationsprogramms bereitgestellt wird, ist die getestete und unterstützte Version. Beachten Sie, dass der `macos` Binärdateien sind im enthalten `/extras` Ein Teil der ZIP-Datei wird nicht getestet oder unterstützt.

Neuerungen in 23.07.1

Kubernetes: Behobene Dämonenlöschung zur Unterstützung von Upgrades ohne Ausfallzeiten ("[Ausgabe #740](#)").

Änderungen in 23.07

Korrekturen

Kubernetes

- Trident Upgrade wurde korrigiert, um alte Pods, die sich im Abschlusszustand befinden, zu ignorieren ("[Ausgabe #740](#)").
- Tolerierung zur Definition „transient-trident-Version-pod“ hinzugefügt ("[Ausgabe #795](#)").

Astra Trident

- ONTAP-ZAPI-Anforderungen wurden behoben, um sicherzustellen, dass die LUN-Seriennummern abgefragt werden, wenn LUN-Attribute zur Identifizierung und Behebung von Ghost-iSCSI-Geräten während der Knotenstagevorgänge abgerufen werden.
- Fehlerbehandlung im Speichertreibercode ("[Ausgabe #816](#)").
- Feste Quota-Größe bei Verwendung von ONTAP-Treibern mit `use-Rest=true`.
- Erstellung von LUN-Klonen in `ontap-san-Economy` wurde korrigiert.
- Informationsfeld veröffentlichen von zurücksetzen `rawDevicePath` Bis `devicePath`; Zusätzliche Logik zum Ausfüllen und Wiederherstellen (in einigen Fällen) `devicePath` Feld.

Vorgestellt Werden

Kubernetes

- Unterstützung für den Import vorbereiteter Snapshots wurde hinzugefügt.
- Minimale Bereitstellung und Dämonset linux-Berechtigungen ("[Ausgabe #817](#)").

Astra Trident

- Es wird kein Statusfeld mehr für „Online“ Volumes und Snapshots gemeldet.

- Aktualisiert den Back-End-Status, wenn das ONTAP-Backend offline ist ("[Probleme #801](#)", "[#543](#)").
- Die LUN-Seriennummer wird während des Workflows „ControllerVolumePublish“ immer abgerufen und veröffentlicht.
- Zusätzliche Logik zur Überprüfung der Seriennummer und Größe des iSCSI Multipath-Geräts hinzugefügt.
- Zusätzliche Überprüfung für iSCSI-Volumes, um sicherzustellen, dass das richtige Multipath-Gerät nicht bereitgestellt wird.

Experimentelle Verbesserung

Unterstützung für NVMe over TCP für den ONTAP-SAN-Treiber wurde um eine technische Vorschau erweitert.

Dokumentation

Viele organisatorische und formatierte Verbesserungen wurden vorgenommen.

Abschreibungen

Kubernetes

- Unterstützung für v1beta1-Snapshots wurde entfernt.
- Unterstützung für Pre-CSI-Volumes und Speicherklassen wurde entfernt.
- Aktualisiertes, mindestens unterstütztes Kubernetes auf 1.22

Änderungen in 23.04



Volume-Trennung für ONTAP-SAN-*-Volumes erzwingen wird nur bei Kubernetes-Versionen mit aktiviertem Non-Graceful Node Shutdown Feature Gate unterstützt. Die Option zum erzwingen der Trennung muss während der Installation mithilfe des aktiviert sein `--enable-force-detach` Flag für das Trident Installationsprogramm.

Korrekturen

- Trident-Operator zur Verwendung von IPv6-localhost für die Installation festgelegt, wenn in Spec angegeben.
- Trident Operator Cluster-Rollenberechtigungen wurden festgelegt, um mit den Bundle-Berechtigungen synchronisiert zu werden ("[Ausgabe #799](#)").
- Problem beim Anhängen von RAW-Block-Volumes auf mehreren Knoten im RWX-Modus behoben.
- Unterstützung von FlexGroup-Klonen und Volume-Import für SMB-Volumes wurde korrigiert.
- Das Problem, dass der Trident Controller nicht sofort heruntergefahren werden konnte, wurde behoben ("[Ausgabe #811](#)").
- Es wurde ein Fix zur Auflistung aller igroup-Namen hinzugefügt, die mit einer angegebenen LUN verbunden sind, die mit `ontap-san-*` Treibern bereitgestellt wurde.
- Korrektur hinzugefügt, um die Ausführung externer Prozesse bis zum Abschluss zu ermöglichen.
- Kompilierungsfehler für s390-Architektur ("[Ausgabe #537](#)").
- Falsche Protokollierungsebene während der Volume-Mount-Vorgänge ("[Ausgabe #781](#)").
- Fehler bei der Assertion des potenziellen Typs ("[Ausgabe #802](#)").

Vorgestellt Werden

- Kubernetes:
 - Unterstützung für Kubernetes 1.27 hinzugefügt.
 - Unterstützung für den Import von LUKS-Volumes wurde hinzugefügt.
 - Zusätzliche Unterstützung für den ReadWriteOncePod PVC-Zugriffsmodus.
 - Unterstützung für Force-Trennen für ONTAP-SAN-* -Volumes während nicht-Graceful Node Shutdown-Szenarien hinzugefügt.
 - Alle ONTAP-SAN-* Volumes verwenden nun Initiatorgruppen pro Node. LUNs werden nur Initiatorgruppen zugeordnet, während sie aktiv auf diesen Nodes veröffentlicht werden, um unsere Sicherheit zu verbessern. Bestehende Volumes werden opportunistisch auf das neue igroup Schema umgestellt, wenn Trident feststellt, dass es sicher ist, dies zu tun, ohne aktive Workloads zu beeinträchtigen ("[Ausgabe #758](#)").
 - Verbesserte die Trident-Sicherheit durch Bereinigung nicht genutzter Trident-gemanagter Initiatorgruppen aus ONTAP-SAN-* Back-Ends.
- Zusätzliche Unterstützung für SMB Volumes mit Amazon FSX für die ontap-nas-Wirtschaft und ontap-nas-flexgroup-Storage-Treiber.
- Unterstützung von SMB-Freigaben mit ontap-nas, ontap-nas-Economy und ontap-nas-Flexgroup-Storage-Treibern hinzugefügt.
- Unterstützung für arm64 Knoten ("[Ausgabe #732](#)").
- Verbessertes Trident Shutdown-Verfahren durch Deaktivieren von API-Servern zuerst ("[Ausgabe #811](#)").
- Cross-Plattform-Build-Unterstützung für Windows- und arm64-Hosts zu Makefile hinzugefügt; siehe BUILD.md.

Abschreibungen

Kubernetes: bei der Konfiguration von ONTAP-san- und ontap-san-Economy-Treibern werden nicht mehr über Back-End-Scoped-Initiatorgruppen erstellt ("[Ausgabe #758](#)").

Änderungen in 23.01.1

Korrekturen

- Trident-Operator zur Verwendung von IPv6-localhost für die Installation festgelegt, wenn in Spec angegeben.
- Die Berechtigungen für die Trident Operator Cluster-Rolle wurden festgelegt, um mit den Bundle-Berechtigungen synchronisiert zu werden ("[Ausgabe #799](#)").
- Korrektur hinzugefügt, um die Ausführung externer Prozesse bis zum Abschluss zu ermöglichen.
- Problem beim Anhängen von RAW-Block-Volumes auf mehreren Knoten im RWX-Modus behoben.
- Unterstützung von FlexGroup-Klonen und Volume-Import für SMB-Volumes wurde korrigiert.

Änderungen in 23.01



Kubernetes 1.27 wird jetzt in Trident unterstützt. Aktualisieren Sie Astra Trident vor dem Upgrade von Kubernetes.

Korrekturen

- Kubernetes: Zusätzliche Optionen zum Ausschließen der Pod-Erstellung von Sicherheitsrichtlinien, um Trident-Installationen über Helm (zu beheben "[Ausgaben #783, #794](#)").

Vorgestellt Werden

Kubernetes

- Zusätzliche Unterstützung für Kubernetes 1.26
- Verbesserung der allgemeinen Trident RBAC-Ressourcenauslastung ("[Ausgabe #757](#)").
- Verbesserte Automatisierung zum Erkennen und Beheben defekter oder veralteter iSCSI Sitzungen auf Host Nodes
- Unterstützung für Erweiterung der LUKS-verschlüsselten Volumes hinzugefügt.
- Kubernetes: Unterstützung für die Rotation von Anmeldeinformationen für LUKS-verschlüsselte Volumes hinzugefügt.

Astra Trident

- Zusätzlicher Support für SMB Volumes mit Amazon FSX für ONTAP für den ontap-nas-Storage-Treiber
- Unterstützung für NTFS-Berechtigungen bei der Verwendung von SMB-Volumes hinzugefügt.
- Zusätzlicher Support für Storage Pools für GCP Volumes mit CVS Service Level.
- Unterstützung für optionale Verwendung von flexgroupAggregateList bei der Erstellung von FlexGroups mit dem ontap-nas-flexgroup Storage-Treiber hinzugefügt.
- Verbesserte Performance für den ontap-nas-Economy-Storage-Treiber beim Management mehrerer FlexVols.
- Aktivierte Daten-LIF-Updates für alle ONTAP-NAS-Speichertreiber.
- Aktualisierte die Namenskonvention für Trident Deployment und DemonSet zur Berücksichtigung des Host-Node-Betriebssystems.

Abschreibungen

- Kubernetes: Aktualisierte die minimal unterstützte Version von Kubernetes auf 1.21.
- Daten-LIFs sollten bei der Konfiguration nicht mehr angegeben werden `ontap-san` Oder `ontap-san-economy` Treiber.

Änderungen in 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 den Astra Trident auf 22.10 aktualisieren.
- 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.

Korrekturen

- Problem wurde speziell mit dem ONTAP Back-End behoben, das mit erstellt wurde `credentials` Feld nicht online während 22.07.0 Upgrade ("[Ausgabe #759](#)").
- **Docker:** hat ein Problem behoben, das dazu führt, dass das Docker Volume Plugin in einigen Umgebungen nicht startet ("[Ausgabe #548](#)" Und "[Ausgabe #760](#)").
- Festes SLM-Problem speziell für ONTAP-SAN-Back-Ends, das sicherstellt, dass nur eine Teilmenge von Daten-LIFs, die zu den Berichterstellungs-Nodes gehören, veröffentlicht wird.
- Es wurde ein Performance-Problem behoben, bei dem unnötige Scans für iSCSI-LUNs beim Anschließen eines Volumes aufgetreten sind.
- Granulare Wiederholungen innerhalb des Astra Trident iSCSI-Workflows entfernt, um schnell zu scheitern und externe Wiederholungsintervalle zu reduzieren.
- Das Problem wurde behoben, bei dem beim Spülen eines iSCSI-Geräts ein Fehler zurückgegeben wurde, als das entsprechende Multipath-Gerät bereits gespült wurde.

Vorgestellt Werden

- Kubernetes:
 - Zusätzliche Unterstützung für Kubernetes 1.25 Vor dem Upgrade auf Kubernetes 1.25 müssen Sie den Astra Trident auf 22.10 aktualisieren.
 - Hinzufügung eines separaten ServiceAccount, ClusterRole und ClusterBinding für die Trident Deployment und DemonSet, um zukünftige Berechtigungsverbesserungen zu ermöglichen.
 - Zusätzlicher Support für "[Namespace-übergreifende Volume-Freigabe](#)".
- Trident Ist Alles `ontap-*` Storage-Treiber arbeiten jetzt mit der ONTAP REST API.
- Neuer Operator yml hinzugefügt (`bundle_post_1_25.yaml`) Ohne A PodSecurityPolicy Die Kubernetes 1.25 unterstützen.
- Hinzugefügt "[Unterstützung für LUKS-verschlüsselte Volumes](#)" Für `ontap-san` Und `ontap-san-economy` Storage-Treiber:
- Unterstützung für Windows Server 2019-Knoten hinzugefügt.
- Hinzugefügt "[Unterstützung für SMB Volumes auf Windows Nodes](#)" Durch die `azure-netapp-files` Storage-Treiber:
- Die automatische MetroCluster-Umschaltermessung für ONTAP-Treiber ist jetzt allgemein verfügbar.

Abschreibungen

- **Kubernetes:** Aktualisiert unterstützt mindestens Kubernetes auf 1.20.
- Astra Data Store (ADS)-Treiber entfernt.
- Unterstützung für `yes` Und `smart` Optionen für `find_multipaths` Wenn Sie Multipathing für Worker-Node für iSCSI konfigurieren.

Änderungen in 22.07

Korrekturen

Kubernetes

- Problem wurde behoben, um boolesche Werte und Zahlenwerte für die Node-Auswahl bei der Konfiguration von Trident mit Helm oder dem Trident Operator zu behandeln. ("[GitHub Ausgabe #700](#)")
- Problem beim Umgang mit Fehlern aus dem nicht-CHAP-Pfad behoben, sodass kubelet erneut versuchen wird, wenn er fehlschlägt. ("[GitHub Ausgabe #736](#)")

Vorgestellt Werden

- Übergang von `k8s.gcr.io` zu `Registry.k8s.io` als Standard-Registry für CSI-Bilder
- ONTAP-SAN Volumes werden jetzt Initiatorgruppen pro Node verwenden und LUNs nur Initiatorgruppen zuordnen, während diese Nodes aktiv veröffentlicht werden, um unsere Sicherheit zu verbessern. Vorhandene Volumes werden opportunistisch auf das neue `igroup`-Schema umgestellt, wenn Astra Trident feststellt, dass es sicher ist, dies ohne aktive Workloads zu beeinträchtigen.
- Enthält eine ResourceQuota mit Trident-Installationen, um sicherzustellen, dass Trident DemonSet geplant ist, wenn die PriorityClass-Nutzung standardmäßig beschränkt ist.
- Unterstützung für Netzwerkfunktionen für den Azure NetApp Files-Treiber hinzugefügt. ("[GitHub Ausgabe #717](#)")
- Technische Vorschau Automatische MetroCluster-Umschalterkennung zu ONTAP-Treibern hinzugefügt. ("[GitHub Ausgabe #228](#)")

Abschreibungen

- **Kubernetes:** Aktualisiert unterstützt mindestens Kubernetes auf 1.19.
- Back-End-Konfiguration ermöglicht nicht mehr mehrere Authentifizierungstypen in einer einzigen Konfiguration.

Umzüge

- Der AWS CVS-Treiber (veraltet seit 22.04) wurde entfernt.
- Kubernetes
 - Keine unnötige `SYS_ADMIN`-Funktion von Node-Pods entfernt.
 - Reduziert die Knotenvorbereitung auf einfache Host-Informationen und aktive Serviceerkennung, um den Aufwand zu reduzieren
Bestätigung, dass NFS/iSCSI-Dienste auf Arbeitsknoten verfügbar sind.

Dokumentation

Eine neue "[Pod-Sicherheitsstandards](#)" (PSS) der Abschnitt enthält Details über Berechtigungen, die von Astra Trident bei der Installation aktiviert wurden.

Änderungen in 22.04

NetApp verbessert seine Produkte und Services kontinuierlich. Im Folgenden finden Sie einige der neuesten Funktionen von Astra Trident: Frühere Versionen finden Sie unter "[Frühere Versionen der Dokumentation](#)".



Wenn Sie ein Upgrade von früheren Trident Versionen durchführen und Azure NetApp Files verwenden, finden Sie das `location` Der Parameter `config` ist jetzt ein Pflichtfeld, `singleton`.

Korrekturen

- Verbessertes Analysieren von iSCSI-Initiatornamen. ("[GitHub Ausgabe #681](#)")
- Das Problem wurde behoben, bei dem CSI-Speicherlassenparameter nicht zulässig waren. ("[GitHub Ausgabe #598](#)")
- Doppelte Schlüsseldeklaration im Trident CRD behoben. ("[GitHub Ausgabe #671](#)")
- Fehlerhafte CSI-Snapshot-Protokolle wurden korrigiert. ("[GitHub Ausgabe #629](#)")
- Problem beim Aufheben der Veröffentlichung von Volumes auf gelöschten Nodes behoben. ("[GitHub Ausgabe #691](#)")
- Zusätzliche Bearbeitung von Inkonsistenzen im Dateisystem auf Blockgeräten. ("[GitHub Ausgabe #656](#)")
- Problem beim Ziehen von Bildern mit automatischer Unterstützung beim Einstellen des behoben `imageRegistry` Markierung während der Installation. ("[GitHub Ausgabe #715](#)")
- Es wurde ein Problem behoben, bei dem der Azure NetApp Files-Treiber ein Volume mit mehreren Exportregeln nicht klonen konnte.

Vorgestellt Werden

- Eingehende Verbindungen zu den sicheren Endpunkten von Trident erfordern jetzt mindestens TLS 1.3. ("[GitHub Ausgabe #698](#)")
- Trident fügt jetzt HSTS-Header zu den Antworten von seinen sicheren Endpunkten hinzu.
- Trident versucht nun, die Azure NetApp Files unix Berechtigungsfunktion automatisch zu aktivieren.
- **Kubernetes:** Trident Demonset wird jetzt in der Klasse mit System-Node-kritischer Priorität ausgeführt. ("[GitHub Ausgabe #694](#)")

Umzüge

E-Series-Treiber (deaktiviert seit 20.07) wurde entfernt.

Änderungen in 22.01.1

Korrekturen

- Problem beim Aufheben der Veröffentlichung von Volumes auf gelöschten Nodes behoben. ("[GitHub Ausgabe #691](#)")
- Fester Panik beim Zugriff auf Nil-Felder für den aggregierten Speicherplatz in den ONTAP API Antworten.

Änderungen in 22.01.0

Korrekturen

- **Kubernetes:** Erhöhung der Neuzulassung der Knotenregistrierung für große Cluster.
- Das Problem wurde behoben, bei dem der Azure-netapp-Files Treiber von mehreren Ressourcen mit demselben Namen verwirrt werden konnte.
- ONTAP SAN IPv6 Daten-LIFs funktionieren jetzt, wenn sie mit Klammern angegeben sind.
- Das Problem wurde behoben, bei dem der Import eines bereits importierten Volumes das EOF zurückgibt, sodass PVC in den ausstehenden Zustand zurückbleibt. ("[GitHub Ausgabe #489](#)")
- Problem behoben, wenn Astra Trident die Performance verlangsamt, wenn > 32 Snapshots auf einem SolidFire Volume erstellt werden.
- SHA-1 wurde durch SHA-256 bei der Erstellung eines SSL-Zertifikats ersetzt.
- Azure NetApp Files-Treiber wurde behoben, um doppelte Ressourcennamen zu erlauben und Vorgänge auf einen einzelnen Speicherort zu beschränken.
- Azure NetApp Files-Treiber wurde behoben, um doppelte Ressourcennamen zu erlauben und Vorgänge auf einen einzelnen Speicherort zu beschränken.

Vorgestellt Werden

- Verbesserungen von Kubernetes:
 - Zusätzliche Unterstützung für Kubernetes 1.23
 - Fügen Sie bei der Installation über Trident Operator oder Helm Planungsoptionen für Trident Pods hinzu. ("[GitHub Ausgabe #651](#)")
- Erlauben Sie regionenübergreifende Volumes im GCP-Treiber. ("[GitHub Ausgabe #633](#)")
- Unterstützung für die Option „unixPermissions“ für Azure NetApp Files Volumes wurde hinzugefügt. ("[GitHub Ausgabe #666](#)")

Abschreibungen

Die Trident REST-Schnittstelle kann nur unter 127.0.0.1 oder [: 1] Adressen zuhören und bedient werden

Änderungen in 21.10.1



In der Version v21.10.0 kann der Trident Controller in den CrashLoopBackOff-Status versetzt werden, wenn ein Node entfernt und dann wieder zum Kubernetes Cluster hinzugefügt wird. Dieses Problem wurde in der Version 21,10,1 behoben (GitHub Ausgabe 669).

Korrekturen

- Beim Import eines Volumes auf ein GCP CVS Backend wurde eine potenzielle Race-Bedingung behoben, die zu einem Import führt.
- Es wurde ein Problem behoben, durch das der Trident Controller in den CrashLoopBackOff-Status versetzt werden kann, wenn ein Node entfernt und dann wieder zum Kubernetes Cluster hinzugefügt wird (GitHub Ausgabe 669).
- Das Problem wurde behoben, bei dem SVMs nicht mehr erkannt wurden, wenn kein SVM-Name angegeben wurde (GitHub Problem 612).

Änderungen in 21.10.0

Korrekturen

- Es wurde ein Problem behoben, bei dem Klone von XFS-Volumes nicht auf demselben Node wie das Quell-Volume gemountet werden konnten (GitHub Ausgabe 514).
- Das Problem wurde behoben, bei dem Astra Trident einen fatalen Fehler beim Herunterfahren protokolliert hat (GitHub Ausgabe 597).
- Kubernetes-bezogene Fixes:
 - Der verwendete Speicherplatz eines Volume wird als Mindestrückstellungsgröße bei der Erstellung von Snapshots mit zurückgegeben `ontap-nas` Und `ontap-nas-flexgroup` Treiber (GitHub Ausgabe 645).
 - Problem behoben wo `Failed to expand filesystem` Fehler wurde nach der Volume-Größe protokolliert (GitHub-Problem 560).
 - Problem behoben, in dem ein POD feststecken konnte `Terminating State` (GitHub Ausgabe 572).
 - Den Fall an der Stelle behoben, an der ein `ontap-san-economy` FlexVol könnte voll von Snapshot-LUNs sein (GitHub Ausgabe 533).
 - Problem mit dem benutzerdefinierten YAML-Installationsprogramm mit einem anderen Bild wurde behoben (GitHub Ausgabe 613).
 - Berechnung der Snapshot-Größe wurde korrigiert (GitHub Ausgabe 611).
 - Das Problem wurde behoben, bei dem alle Astra Trident Installationsprogramme schlicht Kubernetes als OpenShift identifizieren konnten (GitHub Ausgabe 639).
 - Der Trident-Operator hat den Abgleich behoben, wenn der Kubernetes-API-Server nicht erreichbar ist (GitHub Ausgabe 599).

Vorgestellt Werden

- Zusätzlicher Support für `unixPermissions` Option für GCP-CVS Performance Volumes:
- Zusätzliche Unterstützung für für für Skalierung optimierte CVS Volumes in GCP im Bereich von 600 gib bis 1 tib.
- Verbesserungen im Zusammenhang mit Kubernetes:
 - Zusätzliche Unterstützung für Kubernetes 1.22
 - Trident Operator und Helm Chart wurde für die Verwendung mit Kubernetes 1.22 aktiviert (GitHub Ausgabe 628).
 - Bedienerbild zu hinzugefügt `tridentctl` Image-Befehl (GitHub Ausgabe 570).

Experimentelle Verbesserungen

- Zusätzliche Unterstützung für Volume-Replikation im `ontap-san` Treiber.
- Zusätzliche **Tech Preview** REST-Unterstützung für die `ontap-nas-flexgroup`, `ontap-san`, und `ontap-nas-economy` Treiber.

Bekannte Probleme

Bekannte Probleme erkennen Probleme, die eine erfolgreiche Verwendung des Produkts verhindern könnten.

- Wenn Sie ein Kubernetes-Cluster von 1.24 auf 1.25 oder höher aktualisieren, auf das Astra Trident installiert ist, müssen Sie Werte.yaml aktualisieren `excludePodSecurityPolicy` Bis `true` Oder hinzufügen `--set excludePodSecurityPolicy=true` Bis zum `helm upgrade` Befehl bevor Sie ein Upgrade des Clusters durchführen können.
- Astra Trident erzwingt jetzt ein Leereinschub `fsType` (`fsType=""`) Für Volumen, die nicht die haben `fsType` Festgelegt in ihrer StorageClass. Bei der Arbeit mit Kubernetes 1.17 oder höher unterstützt Trident das Ausgeben eines Leerzeichen `fsType` Für NFS-Volumes. Für iSCSI-Volumes müssen Sie die festlegen `fsType` Auf Ihrer StorageClass bei der Durchsetzung eines `fsGroup` Verwenden eines Sicherheitskontexts.
- Wenn Sie ein Backend über mehrere Astra Trident Instanzen hinweg verwenden, sollte jede Back-End-Konfigurationsdatei ein anderes haben `storagePrefix` Für ONTAP-Back-Ends verwenden Sie einen anderen Wert `TenantName` Für SolidFire Back-Ends. Astra Trident kann Volumes nicht erkennen, die andere Instanzen von Astra Trident erstellt haben. Es ist erfolgreich, ein vorhandenes Volume auf ONTAP- oder SolidFire-Back-Ends zu erstellen, da Astra Trident die Volume-Erstellung als einen idempotenten Vorgang behandelt. Wenn `storagePrefix` Oder `TenantName` Unterscheiden sich nicht, es können Namenskonflikte bei Volumes bestehen, die auf demselben Backend erstellt wurden.
- Bei der Installation von Astra Trident (mit `tridentctl` Oder dem Trident Operator) und mit `tridentctl` Für das Management von Astra Trident sollten Sie die sicherstellen `KUBECONFIG` Umgebungsvariable wird festgelegt. Dies ist erforderlich, um für den Kubernetes-Cluster anzugeben `tridentctl` Sollten gegenarbeiten. Bei der Arbeit mit mehreren Kubernetes-Umgebungen sollten Sie sicherstellen, dass die `KUBECONFIG` Die Datei wird genau stammt.
- Um Online-Speicherplatzrückgewinnung für iSCSI PVS durchzuführen, muss das zugrunde liegende Betriebssystem auf dem Worker-Node möglicherweise Mount-Optionen an das Volume übergeben werden. Dies gilt für RHEL/RedHat CoreOS Instanzen, die die benötigen `discard` "[Mount-Option](#)"; Stellen Sie sicher, dass die `MountOption` von der Karte in Ihrem enthalten ist[`StorageClass^`] unterstützt das Online-Blockabwerfen.
- Wenn für den Kubernetes Cluster mehr als eine Instanz von Astra Trident zur Verfügung steht, kann Astra Trident nicht mit anderen Instanzen kommunizieren und kann nicht andere Volumes ermitteln, die sie erstellt haben. Dies führt zu einem unerwarteten und falschen Verhalten, wenn mehrere Instanzen innerhalb eines Clusters ausgeführt werden. Astra Trident sollte nur eine Instanz pro Kubernetes Cluster geben.
- Bei Astra Trident-basiert `StorageClass` Die Objekte werden aus Kubernetes gelöscht, während Astra Trident offline ist, entfernt Astra Trident nicht die entsprechenden Storage-Klassen aus seiner Datenbank, wenn sie wieder online kommt. Sie sollten diese Speicherklassen mit löschen `tridentctl` Oder DIE REST API.
- Wenn ein Benutzer ein von Astra Trident bereitgestelltes PV löscht, bevor das entsprechende PVC gelöscht wird, löscht Astra Trident nicht automatisch das Back-Volume. Sie sollten die Lautstärke über entfernen `tridentctl` Oder DIE REST API.
- ONTAP kann nicht gleichzeitig mehr als ein FlexGroup gleichzeitig bereitstellen, es sei denn, der Satz der Aggregate ist auf jede Bereitstellungsanforderung beschränkt.
- Bei der Verwendung von Astra Trident über IPv6 sollten Sie angeben `managementLIF` Und `dataLIF` In der Back-End-Definition in eckigen Klammern. Beispiel:
`[fd20:8b1e:b258:2000:f816:3eff:feec:0]`.



Sie können nicht angeben `dataLIF` Auf einem ONTAP-SAN-Back-End Astra Trident erkennt alle verfügbaren iSCSI LIFs und erstellt mit ihnen die Multipath-Sitzung.

- Wenn Sie das verwenden `solidfire-san` Treiber mit OpenShift 4.5, stellen Sie sicher, dass die

zugrunde liegenden Worker-Knoten MD5 als CHAP-Authentifizierungsalgorithmus verwenden. Sichere, FIPS-konforme CHAP-Algorithmen SHA1, SHA-256 und SHA3-256 sind mit Element 12.7 erhältlich.

Weitere Informationen

- ["Astra Trident GitHub"](#)
- ["Astra Trident Blogs"](#)

Frühere Versionen der Dokumentation

Mit der Versionsauswahl können Sie zwischen der aktuellen Version und den vorherigen vier Versionen navigieren oder die untenstehenden Links verwenden.

Frühere Versionen

Wenn Sie Astra Trident 23.07 nicht ausführen, ist die Dokumentation für frühere Versionen auf Basis der verfügbar ["Astra Trident Support-Lebenszyklus"](#).

- ["Astra Trident 23.04"](#)
- ["Astra Trident 23.01"](#)
- ["Astra Trident 22.10"](#)
- ["Astra Trident 22.07"](#)
- ["Astra Trident 22.04"](#)
- ["Astra Trident 22.01"](#)
- ["Astra Trident 21.10"](#)
- ["Astra Trident 21.07"](#)

Los geht's

Erfahren Sie mehr über Astra Trident

Erfahren Sie mehr über Astra Trident

Astra Trident ist ein vollständig unterstütztes Open-Source-Projekt, das von NetApp als Teil der gepflegt wird ["Astra Produktfamilie"](#). Es wurde entwickelt, damit Sie die Persistenz-Anforderungen Ihrer Container-Applikation mithilfe von Standardschnittstellen, wie dem Container Storage Interface (CSI), erfüllen können.

Was ist Astra?

Astra erleichtert Unternehmen das Management, die Sicherung und das Verschieben ihrer datenintensiven Container-Workloads, die auf Kubernetes ausgeführt werden, innerhalb der Public Cloud und vor Ort.

Astra stellt persistenten Container-Storage auf Basis von Astra Trident bereit und bietet diese an. Es bietet außerdem erweiterte applikationsgerechte Datenmanagement-Funktionen wie Snapshot, Backup und Restore, Aktivitätsprotokolle und aktives Klonen für Datensicherung, Disaster/Daten-Recovery, Datenaudit und Migrationsanwendungsfälle für Kubernetes-Workloads.

Weitere Informationen zu ["Astra oder melden Sie sich für die kostenlose Testversion an"](#).

Was ist Astra Trident?

Astra Trident ermöglicht die Nutzung und das Management von Storage-Ressourcen über alle gängigen NetApp Storage-Plattformen hinweg, in der Public Cloud oder lokal, einschließlich ONTAP (AFF, FAS, Select, Cloud, Amazon FSX for NetApp ONTAP), Element Software (NetApp HCI, SolidFire), Azure NetApp Files Service und Cloud Volumes Service auf Google Cloud.

Astra Trident ist ein CSI-konformer dynamischer Storage Orchestrator, der sich nativ integrieren lässt ["Kubernetes"](#). Astra Trident wird als einzelner Controller Pod plus Node Pod auf jedem Worker-Node im Cluster ausgeführt. Siehe ["Die Architektur von Astra Trident"](#) Entsprechende Details.

Astra Trident bietet zudem eine direkte Integration in das Docker Ecosystem für NetApp Storage-Plattformen. Das NetApp Docker Volume Plug-in (nDVP) unterstützt die Bereitstellung und das Management von Storage-Ressourcen von der Storage-Plattform an Docker Hosts. Siehe ["Implementieren Sie Astra Trident für Docker"](#) Entsprechende Details.



Wenn Sie Kubernetes zum ersten Mal verwenden, sollten Sie sich mit dem vertraut machen ["Kubernetes-Konzepte und -Tools"](#).

Machen Sie einen Testlauf mit Astra Trident

Fordern Sie für einen Testlauf den Zugriff auf den „persistenten Storage für Container-Workloads einfach implementieren und klonen“ an. ["NetApp Testversion"](#) Verwenden eines gebrauchsfertigen Laborabbaus. Testlauf bietet eine Sandbox-Umgebung mit einem Kubernetes-Cluster mit drei Nodes und Astra Trident ist installiert und konfiguriert. So können Sie sich besser mit Astra Trident vertraut machen und die zugehörigen Funktionen erkunden.

Eine weitere Option ist die ["Installationsanleitung für kubeadm"](#) Von Kubernetes bereitgestellt.



Verwenden Sie in einer Produktionsumgebung keine Kubernetes-Cluster, die Sie mit diesen Anweisungen erstellen. Nutzen Sie die von Ihrer Distribution bereitgestellten Leitfäden zur Implementierung in Produktionsumgebungen für Cluster.

Finden Sie weitere Informationen

- ["Die NetApp Astra-Produktfamilie"](#)
- ["Dokumentation des Astra Control Service"](#)
- ["Astra Control Center-Dokumentation"](#)
- ["Astra API-Dokumentation"](#)

Die Architektur von Astra Trident

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

Allgemeines zu Controller-Pods und Node-Pods

Astra Trident wird einzeln implementiert [Trident Controller Pod](#) Und eines oder mehrere [Trident Node Pods](#) Auf dem Kubernetes-Cluster und verwendet standardmäßige Kubernetes [CSI Sidecar Container](#), um die Implementierung von CSI-Plug-ins zu vereinfachen. ["Kubernetes CSI Sidecar-Container"](#) Werden von der Kubernetes Storage Community unterhalten.

Kubernetes ["Knotenauswahl"](#) Und ["Toleranzen und Verfleckungen"](#) Werden verwendet, um die Ausführung eines Pod auf einem bestimmten oder bevorzugten Node einzuschränken. Während der Astra Trident Installation können Node-Selektoren und Toleranzen für Controller- und Node-Pods konfiguriert werden.

- 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.

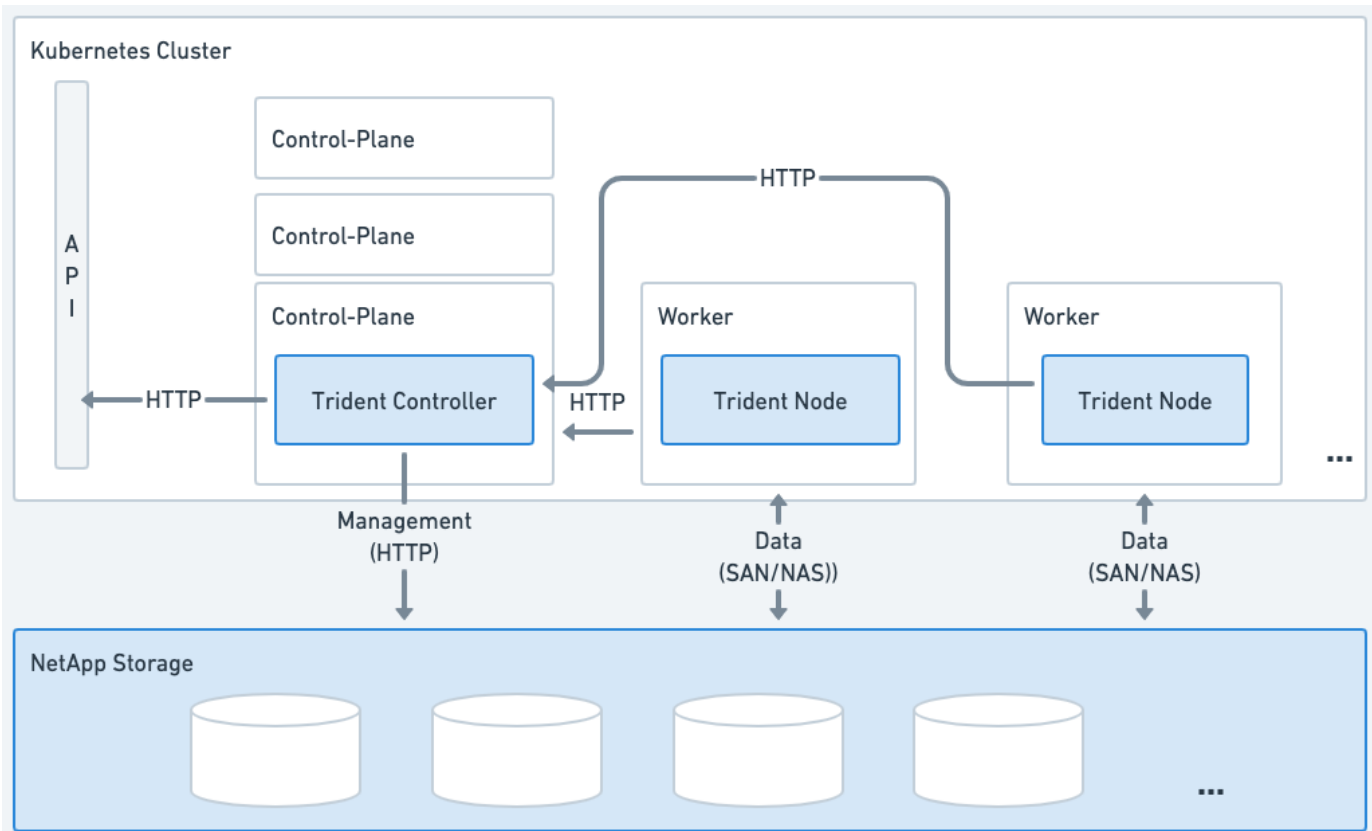


Abbildung 1. Astra Trident wird auf dem Kubernetes-Cluster implementiert

Trident Controller Pod

Beim Trident Controller Pod handelt es sich um einen einzelnen Pod, auf dem das CSI Controller Plug-in ausgeführt wird.

- Verantwortlich für die Bereitstellung und das Management von Volumes in NetApp Storage
- Management durch eine Kubernetes-Implementierung
- Kann je nach Installationsparameter auf der Steuerebene oder auf den Arbeitsknoten ausgeführt werden.

Abbildung 2. Trident Controller Pod-Diagramm

Trident Node Pods

Trident Node Pods sind privilegierte Pods, auf denen das CSI Node Plug-in ausgeführt wird.

- Verantwortlich für das Mounten und Entmounten von Speicher für Pods, die auf dem Host ausgeführt werden
- Gemanagt von einem Kubernetes DemonSet
- Muss auf jedem Node ausgeführt werden, auf dem NetApp Storage gemountet werden soll

Abbildung 3. Trident Node Pod-Diagramm

Unterstützte Kubernetes-Cluster-Architekturen

Astra Trident wird durch die folgenden Kubernetes-Architekturen unterstützt:

| Kubernetes-Cluster-Architekturen | Unterstützt | Standardinstallation |
|--|-------------|----------------------|
| Ein Master Computing | Ja. | Ja. |
| Mehrere Master-Computer und Computing-Ressourcen | Ja. | Ja. |
| Master, etcd, Datenverarbeitung | Ja. | Ja. |
| Master, Infrastruktur, Computing | Ja. | Ja. |

Konzepte

Bereitstellung

Die Bereitstellung in Astra Trident besteht aus zwei Hauptphasen. In der ersten Phase wird eine Speicherklasse mit einem Satz geeigneter Back-End-Speicherpools verknüpft. Diese werden vor der Bereitstellung als notwendig vorbereitet. Die zweite Phase umfasst die Volume-Erstellung selbst und erfordert die Auswahl eines Speicherpools aus denen, die mit der Storage-Klasse des ausstehenden Volumes verknüpft sind.

Storage-Klassen-Zuordnung

Das Zuordnen von Back-End Storage-Pools zu einer Storage-Klasse hängt sowohl von den angeforderten Attributen der Storage-Klasse als auch von deren `storagePools`, `additionalStoragePools`, und `excludeStoragePools` Listen. Wenn Sie eine Storage-Klasse erstellen, vergleicht Trident die von jedem seiner Back-Ends angebotenen Attribute und Pools mit den von der Storage-Klasse angeforderten Attributen. Wenn die Attribute und der Name eines Storage Pools mit allen angeforderten Attributen und Pool-Namen übereinstimmen, fügt Astra Trident diesem Satz an geeigneten Storage-Pools für diese Storage-Klasse hinzu. Außerdem fügt Astra Trident alle im aufgeführten Storage-Pools hinzu `additionalStoragePools` Listen Sie zu diesem Satz auf, auch wenn seine Attribute nicht alle oder eines der angeforderten Attribute der Storage-Klasse erfüllen. Sie sollten das verwenden `excludeStoragePools` Liste zum Überschreiben und Entfernen von Speicherpools, die für eine Speicherklasse verwendet werden. Astra Trident führt jedes Mal einen ähnlichen Prozess durch, wenn Sie ein neues Back-End hinzufügen. Er überprüft, ob die Storage Pools die Anforderungen der vorhandenen Storage-Klassen erfüllen und entfernt alle, die als ausgeschlossen markiert wurden.

Volume-Erstellung

Astra Trident verwendet dann die Zuordnungen zwischen Storage-Klassen und Storage-Pools, um zu bestimmen, wo Volumes bereitgestellt werden sollen. Wenn Sie ein Volume erstellen, erhält Astra Trident zunächst die Reihe von Storage-Pools für dieses Volume in der Storage-Klasse. Wenn Sie ein Protokoll für das Volume angeben, entfernt Astra Trident die Storage-Pools, die das angeforderte Protokoll nicht bereitstellen können (beispielsweise kann ein NetApp HCI/SolidFire Backend kein dateibasiertes Volume bereitstellen, während ein ONTAP NAS-Backend kein blockbasiertes Volume bereitstellen kann). Astra Trident randomisiert die Reihenfolge dieser daraus resultierenden Sets, um eine gleichmäßige Verteilung der Volumes zu ermöglichen und es anschließend zu iterieren und dabei zu versuchen, das Volume wiederum auf jedem Storage-Pool bereitzustellen. Wenn sie erfolgreich ist, wird sie erfolgreich zurückgegeben, und es werden alle Fehler protokolliert, die im Prozess aufgetreten sind. Astra Trident gibt einen Fehler zurück **nur wenn** sie nicht auf allen * den Storage Pools zur Verfügung steht für die angeforderte Storage-Klasse und das gewünschte Protokoll.

Volume Snapshots

Erfahren Sie mehr darüber, wie Astra Trident die Erstellung von Volume-Snapshots für seine Treiber steuert.

Erfahren Sie mehr über die Erstellung von Volume Snapshots

- Für das `ontap-nas`, `ontap-san`, `gcp-cvs`, und `azure-netapp-files` Treiber, wird jedes Persistent Volume (PV) einer FlexVol zugeordnet. Volume Snapshots werden im Ergebnis als NetApp Snapshots erstellt. NetApp Snapshots liefern weitaus mehr Stabilität, Skalierbarkeit, Wiederherstellbarkeit und Performance als vergleichbare Systeme. Diese Snapshot-Kopien sind äußerst schnell und platzsparend, da sie erstellt und gespeichert werden müssen.
- Für das `ontap-nas-flexgroup` Treiber: Jedes Persistent Volume (PV) ist einem FlexGroup zugeordnet. Im Ergebnis werden Volume Snapshots als NetApp FlexGroup Snapshots erstellt. NetApp Snapshots liefern weitaus mehr Stabilität, Skalierbarkeit, Wiederherstellbarkeit und Performance als vergleichbare Systeme. Diese Snapshot-Kopien sind äußerst schnell und platzsparend, da sie erstellt und gespeichert werden müssen.
- Für das `ontap-san-economy` Treiber, PVS werden LUNs zugeordnet, die auf gemeinsam genutzten FlexVols erstellt wurden. VolumeSnapshots von PVS werden durch FlexClones der zugehörigen LUN erreicht. Mit der ONTAP FlexClone Technologie ist es nahezu sofort möglich, Kopien selbst von größten Datensätzen zu erstellen. Kopien nutzen Datenblöcke gemeinsam mit ihren Eltern und verbrauchen somit keinen Storage, außer was für Metadaten erforderlich ist.
- Für das `solidfire-san` Treiber: Jedes PV wird einer auf der NetApp Element Software/dem NetApp HCI Cluster erstellten LUN zugeordnet. VolumeSnapshots werden durch Element Snapshots der zugrunde liegenden LUN dargestellt. Diese Snapshots sind zeitpunktgenaue Kopien, die nur eine kleine Menge an Systemressourcen und Platz beanspruchen.
- Bei der Arbeit mit dem `ontap-nas` Und `ontap-san` Treiber, ONTAP Snapshots sind zeitpunktgenaue Kopien der FlexVol und verbrauchen Platz auf der FlexVol selbst. Das kann dazu führen, dass der beschreibbare Speicherplatz auf dem Volume mit der Zeit verkürzt wird, wenn Snapshots erstellt/geplant werden. Eine einfache Möglichkeit dieser Bewältigung ist, das Volumen durch die Anpassung über Kubernetes zu vergrößern. Eine weitere Option ist das Löschen von nicht mehr benötigten Snapshots. Wenn ein über Kubernetes erstellter VolumeSnapshot gelöscht wird, löscht Astra Trident den zugehörigen ONTAP-Snapshot. ONTAP Snapshots, die nicht über Kubernetes erstellt wurden, können auch gelöscht werden.

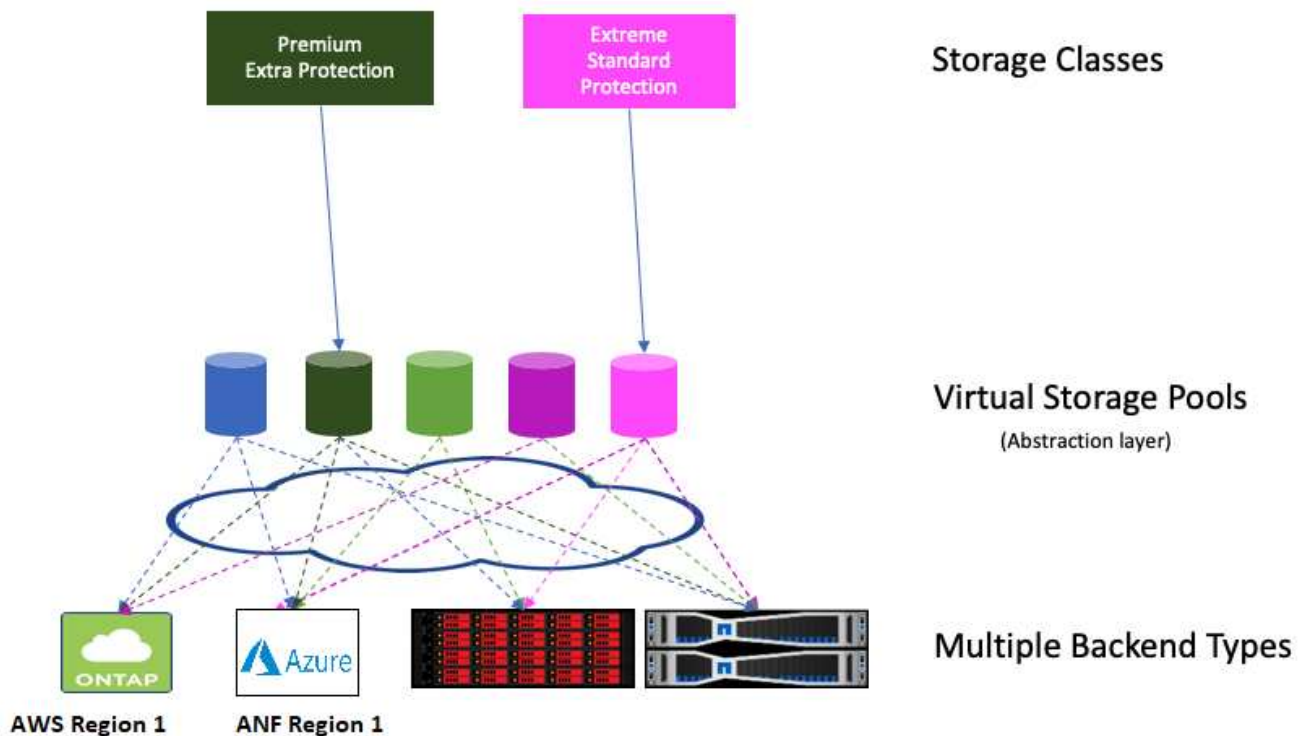
Mit Astra Trident können Sie VolumeSnapshot verwenden, um neue PVS daraus zu erstellen. Die Erstellung von PVS aus diesen Snapshots wird mithilfe der FlexClone Technologie für unterstützte ONTAP- und CVS-Back-Ends durchgeführt. Wenn ein PV aus einem Snapshot erstellt wird, ist das Back-Volume ein FlexClone des übergeordneten Volume des Snapshots. Der `solidfire-san` Der Treiber verwendet Volume Clones der Element Software, um PVS aus Snapshots zu erstellen. Hier erstellt es aus dem Element Snapshot einen Klon.

Virtuelle Pools

Virtuelle Pools stellen eine Abstraktionsschicht zwischen Astra Trident Storage-Back-Ends und Kubernetes bereit `StorageClasses`. Sie ermöglichen es Administratoren, für jedes Back-End-System Aspekte wie Standort, Performance und Schutz zu definieren, ohne dafür eine `StorageClass` Legen Sie fest, welches physische Backend-, Backend-Pool- oder Backend-Typ für die gewünschten Kriterien verwendet werden soll.

Erfahren Sie mehr über virtuelle Pools

Der Storage-Administrator kann virtuelle Pools auf einem beliebigen Astra Trident Back-End in einer JSON- oder YAML-Definitionsdatei definieren.



Jeder außerhalb der Liste der virtuellen Pools angegebene Aspekt ist global für das Backend und gilt für alle virtuellen Pools, während jeder virtuelle Pool einen oder mehrere Aspekte einzeln angeben kann (alle Backend-globalen Aspekte außer Kraft setzen).



- Versuchen Sie beim Definieren virtueller Pools nicht, die Reihenfolge vorhandener virtueller Pools in einer Backend-Definition neu anzuordnen.
- Wir empfehlen, Attribute für einen vorhandenen virtuellen Pool zu ändern. Sie sollten einen neuen virtuellen Pool definieren, um Änderungen vorzunehmen.

Die meisten Aspekte werden Backend-spezifisch angegeben. Entscheidend ist, dass die Aspect-Werte nicht außerhalb des Back-End-Treibers angezeigt werden und nicht für die Abstimmung in verfügbar sind `StorageClasses`. Stattdessen definiert der Administrator eine oder mehrere Labels für jeden virtuellen Pool. Jedes Etikett ist ein Schlüssel:Wert-Paar, und Etiketten können häufig über eindeutige Back-Ends hinweg verwendet werden. Wie Aspekte können auch Labels pro Pool oder global zum Backend angegeben werden. Im Gegensatz zu Aspekten, die vordefinierte Namen und Werte haben, hat der Administrator volle Entscheidungsbefugnis, Beschriftungsschlüssel und -Werte nach Bedarf zu definieren. Storage-Administratoren können Labels je virtuellen Pool definieren und Volumes nach Label gruppieren.

A `StorageClass` identifiziert den virtuellen Pool, der verwendet werden soll, indem auf die Beschriftungen in einem Auswahlparameter Bezug gesetzt wird. Virtuelle Pool-Selektoren unterstützen folgende Operatoren:

| Operator | Beispiel | Der Wert für die Bezeichnung eines Pools muss: |
|----------|-----------------------------------|--|
| = | Performance=Premium | Übereinstimmung |
| != | Performance!=extrem | Keine Übereinstimmung |
| in | Lage in (Osten, Westen) | Werden Sie im Satz von Werten |
| notin | Performance-Dose (Silber, Bronze) | Nicht im Wertungsset sein |
| <key> | Darstellt | Mit einem beliebigen Wert existieren |
| !<key> | !Schutz | Nicht vorhanden |

Volume-Zugriffsgruppen

Erfahren Sie mehr über die Einsatzmöglichkeiten von Astra Trident "[Volume-Zugriffsgruppen](#)".



Ignorieren Sie diesen Abschnitt, wenn Sie CHAP verwenden. Dies wird empfohlen, um die Verwaltung zu vereinfachen und die unten beschriebene Skalierungsgrenze zu vermeiden. Wenn Sie Astra Trident im CSI-Modus verwenden, können Sie diesen Abschnitt ignorieren. Astra Trident verwendet CHAP, wenn es als erweiterte CSI-bereitstellung installiert ist.

Erfahren Sie mehr über Volume Access Groups

Astra Trident kann über Volume-Zugriffsgruppen den Zugriff auf die Volumes steuern, die es bereitstellt. Wenn CHAP deaktiviert ist, wird erwartet, dass eine Zugriffsgruppe mit dem Namen gefunden wird `trident`. Es sei denn, Sie geben eine oder mehrere Zugriffsgruppen-IDs in der Konfiguration an.

Astra Trident ordnet den konfigurierten Zugriffsgruppen neue Volumes zu, allerdings werden dafür keine Zugriffsgruppen selbst erstellt oder anderweitig gemanagt. Die Zugriffsgruppen müssen vorhanden sein, bevor das Storage-Back-End zu Astra Trident hinzugefügt wird. Sie müssen die iSCSI-IQNs von jedem Node im Kubernetes-Cluster enthalten, der die über dieses Back-End bereitgestellten Volumes mounten kann. In den meisten Installationen umfasst dies alle Worker Nodes im Cluster.

Bei Kubernetes-Clustern mit mehr als 64 Nodes sollten Sie mehrere Zugriffsgruppen verwenden. Jede Zugriffsgruppe kann bis zu 64 IQNs enthalten, und jedes Volume kann zu vier Zugriffsgruppen gehören. Bei maximal vier Zugriffsgruppen kann jeder Node in einem Cluster mit einer Größe von bis zu 256 Nodes auf beliebige Volumes zugreifen. Aktuelle Grenzwerte für Volume-Zugriffsgruppen finden Sie unter "[Hier](#)".

Wenn Sie die Konfiguration von einer Konfiguration ändern, die den Standard verwendet `trident` Zugriffsgruppe für eine Gruppe, die auch andere verwendet, geben Sie die ID für die ein `trident` Zugriffsgruppe in der Liste.

Der Einstieg in Astra Trident ist schnell möglich

Sie können Astra Trident installieren und mit dem Management von Storage-Ressourcen in wenigen Schritten beginnen. Bevor Sie beginnen, überprüfen Sie "[Anforderungen von Astra Trident](#)".



Informationen zu Docker finden Sie unter "[Astra Trident für Docker](#)".

1

Installation Von Astra Trident

Astra Trident bietet mehrere Installationsmethoden und -Modi, die für eine Vielzahl von Umgebungen und Organisationen optimiert sind.

["Installation Von Astra Trident"](#)

2

Bereiten Sie den Knoten „Worker“ vor

Alle Worker-Nodes im Kubernetes-Cluster müssen in der Lage sein, die Volumes, die Sie für Ihre Pods bereitgestellt haben, zu mounten.

["Bereiten Sie den Knoten „Worker“ vor"](#)

3

Erstellen Sie ein Backend

Ein Backend definiert die Beziehung zwischen Astra Trident und einem Storage-System. Er erzählt Astra Trident, wie man mit diesem Storage-System kommuniziert und wie Astra Trident Volumes darauf bereitstellen sollte.

["Konfigurieren Sie ein Backend"](#) Für Ihr Speichersystem

4

Kubernetes StorageClass erstellen

Das Objekt Kubernetes StorageClass gibt Astra Trident als bereitstellung an und ermöglicht die Erstellung einer Storage-Klasse zur Bereitstellung von Volumes mit anpassbaren Attributen. Astra Trident erstellt eine passende Storage-Klasse für Kubernetes-Objekte, mit der der Astra Trident-bereitstellung angegeben wird.

["Erstellen Sie eine Speicherklasse"](#)

5

Bereitstellen eines Volumes

Ein *PersistentVolume* (PV) ist eine physische Speicherressource, die vom Cluster-Administrator auf einem Kubernetes-Cluster bereitgestellt wird. Das *PersistentVolumeClaim* (PVC) ist eine Anforderung für den Zugriff auf das PersistentVolume auf dem Cluster.

Erstellen Sie ein PersistentVolume (PV) und ein PersistentVolumeClaim (PVC), das die konfigurierte Kubernetes StorageClass verwendet, um Zugriff auf das PV anzufordern. Anschließend können Sie das PV an einem Pod montieren.

["Bereitstellen eines Volumes"](#)

Was kommt als Nächstes?

Sie können nun zusätzliche Back-Ends hinzufügen, Storage-Klassen managen, Back-Ends managen und Volume-Operationen durchführen.

Anforderungen

Vor der Installation von Astra Trident sollten Sie diese allgemeinen Systemanforderungen überprüfen. Spezifische Back-Ends können zusätzliche Anforderungen haben.

Kritische Informationen zu Astra Trident

Sie müssen die folgenden wichtigen Informationen über Astra Trident lesen.

Informationen über Astra TriperelT

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

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

Unterstützte Frontends (Orchestrators)

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

- Anthos On-Premises (VMware) und Anthos auf Bare Metal 1.12
- Kubernetes 1.22–1.27
- Mirantis Kubernetes Engine 3.5
- OpenShift 4.10 - 4.13

Der Trident-Operator wird durch folgende Versionen unterstützt:

- Anthos On-Premises (VMware) und Anthos auf Bare Metal 1.12
- Kubernetes 1.22–1.27
- OpenShift 4.10 - 4.13

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.



Bevor Sie ein Kubernetes Cluster von 1.24 auf 1.25 oder höher aktualisieren, auf die Astra Trident installiert ist, finden Sie unter "[Aktualisieren einer Helm-Installation](#)".

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 9.5 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? |
|------------------------------|--------------------|-----------------------------|
| Astra Trident | 1.22 - 1.27 | Nein |
| Volume Snapshots | 1.22 - 1.27 | Nein |
| PVC aus Volume Snapshots | 1.22 - 1.27 | Nein |
| ISCSI PV-Größe | 1.22 - 1.27 | Nein |
| Bidirektionales ONTAP-CHAP | 1.22 - 1.27 | Nein |
| Dynamische Exportrichtlinien | 1.22 - 1.27 | Nein |
| Trident Operator | 1.22 - 1.27 | Nein |
| CSI-Topologie | 1.22 - 1.27 | Nein |

Getestete Host-Betriebssysteme

Der Astra Trident unterstützt zwar bestimmte Betriebssysteme offiziell nicht, doch ist es bekannt, dass folgende Betriebssysteme funktionieren:

- Redhat CoreOS (RHCOS) Versionen, die von OpenShift Container Platform (AMD64 und ARM64) unterstützt werden
- RHEL 8 ODER HÖHER (AMD64 UND ARM64)
- Ubuntu 22.04 oder höher (AMD64 und ARM64)
- Windows Server 2019 (AMD64)

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

Alle Worker-Nodes im Kubernetes-Cluster müssen in der Lage sein, die Volumes, die Sie für Ihre Pods bereitgestellt haben, zu mounten. Um die Worker-Knoten vorzubereiten, müssen Sie die NFS- oder iSCSI-Tools auf der Grundlage Ihrer Treiberauswahl installieren.

["Bereiten Sie den Knoten „Worker“ vor"](#)

Konfiguration des Storage-Systems

Astra Trident erfordert möglicherweise Änderungen an einem Storage-System, bevor es mit einer Backend-Konfiguration verwendet werden kann.

["Back-Ends konfigurieren"](#)

Astra Trident-Ports

Astra Trident erfordert Zugriff auf spezifische Ports für die Kommunikation.

["Astra Trident-Ports"](#)

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 |
|--------------------|--|
| V1.22.0 | <ul style="list-style-type: none">• <code>docker.io/netapp/Trident:23.07.1</code>• <code>docker.io/netapp/Trident-AutoSupport:23.07</code>• <code>Registry.k8s.io/SIG-Storage/csi-provisioner:v3.5.0</code>• <code>Registry.k8s.io/SIG-Storage/csi-Attacher:v4.3.0</code>• <code>Registry.k8s.io/SIG-Storage/csi-resizer:v1.8.0</code>• <code>Registry.k8s.io/SIG-Storage/csi-snapshotter:v6.2.2</code>• <code>Registry.k8s.io/SIG-Storage/csi-node-driver-Registrar:v2.8.0</code>• <code>docker.io/netapp/Trident-Operator:23.07.1</code> (optional) |

| Kubernetes-Version | Container-Image |
|--------------------|--|
| V1.23.0 | <ul style="list-style-type: none"> • docker.io/netapp/Trident:23.07.1 • docker.io/netapp/Trident-AutoSupport:23.07 • Registry.k8s.io/SIG-Storage/csi-provisioner:v3.5.0 • Registry.k8s.io/SIG-Storage/csi-Attacher:v4.3.0 • Registry.k8s.io/SIG-Storage/csi-resizer:v1.8.0 • Registry.k8s.io/SIG-Storage/csi-snapshotter:v6.2.2 • Registry.k8s.io/SIG-Storage/csi-Node-driver-Registrar:v2.8.0 • docker.io/netapp/Trident-Operator:23.07.1 (optional) |
| V1.24.0 | <ul style="list-style-type: none"> • docker.io/netapp/Trident:23.07.1 • docker.io/netapp/Trident-AutoSupport:23.07 • Registry.k8s.io/SIG-Storage/csi-provisioner:v3.5.0 • Registry.k8s.io/SIG-Storage/csi-Attacher:v4.3.0 • Registry.k8s.io/SIG-Storage/csi-resizer:v1.8.0 • Registry.k8s.io/SIG-Storage/csi-snapshotter:v6.2.2 • Registry.k8s.io/SIG-Storage/csi-Node-driver-Registrar:v2.8.0 • docker.io/netapp/Trident-Operator:23.07.1 (optional) |
| V1.25.0 | <ul style="list-style-type: none"> • docker.io/netapp/Trident:23.07.1 • docker.io/netapp/Trident-AutoSupport:23.07 • Registry.k8s.io/SIG-Storage/csi-provisioner:v3.5.0 • Registry.k8s.io/SIG-Storage/csi-Attacher:v4.3.0 • Registry.k8s.io/SIG-Storage/csi-resizer:v1.8.0 • Registry.k8s.io/SIG-Storage/csi-snapshotter:v6.2.2 • Registry.k8s.io/SIG-Storage/csi-Node-driver-Registrar:v2.8.0 • docker.io/netapp/Trident-Operator:23.07.1 (optional) |

| Kubernetes-Version | Container-Image |
|--------------------|---|
| V1.26.0 | <ul style="list-style-type: none"> • docker.io/netapp/Trident:23.07.1 • docker.io/netapp/Trident-AutoSupport:23.07 • Registry.k8s.io/SIG-Storage/csi-provisioner:v3.5.0 • Registry.k8s.io/SIG-Storage/csi-Attacher:v4.3.0 • Registry.k8s.io/SIG-Storage/csi-resizer:v1.8.0 • Registry.k8s.io/SIG-Storage/csi-snapshotter:v6.2.2 • Registry.k8s.io/SIG-Storage/csi-node-driver-Registrar:v2.8.0 • docker.io/netapp/Trident-Operator:23.07.1 (optional) |
| V1.27.0 | <ul style="list-style-type: none"> • docker.io/netapp/Trident:23.07.1 • docker.io/netapp/Trident-AutoSupport:23.07 • Registry.k8s.io/SIG-Storage/csi-provisioner:v3.5.0 • Registry.k8s.io/SIG-Storage/csi-Attacher:v4.3.0 • Registry.k8s.io/SIG-Storage/csi-resizer:v1.8.0 • Registry.k8s.io/SIG-Storage/csi-snapshotter:v6.2.2 • Registry.k8s.io/SIG-Storage/csi-node-driver-Registrar:v2.8.0 • docker.io/netapp/Trident-Operator:23.07.1 (optional) |

Installation Von Astra Trident

Erfahren Sie mehr über die Installation von Astra Trident

Damit Astra Trident in einer Vielzahl von Umgebungen und Organisationen installiert werden kann, bietet NetApp diverse Installationsoptionen an. Sie können Astra Trident über den Trident-Operator (manuell oder mit Helm) oder mit `tridentctl` installieren. In diesem Thema finden Sie wichtige Informationen zur Auswahl des richtigen Installationsprozesses.

Kritische Informationen zu Astra Trident 23.07

Sie müssen die folgenden wichtigen Informationen über Astra Trident lesen.

Informationen über Astra Trident

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

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

Bevor Sie beginnen

Unabhängig von Ihrem Installationspfad müssen Sie Folgendes haben:

- Vollständige Berechtigungen für einen unterstützten Kubernetes-Cluster, auf dem eine unterstützte Version von Kubernetes und aktivierte Funktionsanforderungen ausgeführt werden. Überprüfen Sie die ["Anforderungen"](#) Entsprechende Details.
- 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 auf die Kubernetes-Cluster-Konfiguration verweisen.
- Bei Verwendung von Kubernetes mit Docker Enterprise ["Führen Sie die entsprechenden Schritte aus, um den CLI-Zugriff zu aktivieren"](#).



Wenn Sie sich nicht mit dem vertraut gemacht haben ["Grundkonzepte"](#), Ist jetzt eine tolle Zeit, um das zu tun.

Wählen Sie Ihre Installationsmethode

Wählen Sie die für Sie richtige Installationsmethode aus. Sie sollten auch die Überlegungen zu prüfen "[Bewegen zwischen Methoden](#)" Bevor Sie Ihre Entscheidung treffen.

Verwenden des Betreibers von Trident

Ob manuell oder mit Hilfe von Helm – der Trident Operator ist ein hervorragender Weg, die Installation zu vereinfachen und Astra Trident Ressourcen dynamisch zu managen. Das können Sie sogar "[Individuelle Anpassung der Trident Implementierung](#)" Verwenden der Attribute im `TridentOrchestrator` Benutzerdefinierte Ressource (CR).

Die Vorteile der Verwendung des Trident-Mitarbeiters:

** für Objekte aus Trident**

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

- Servicekonto für den Betreiber
- ClusterRole und ClusterRoleBinding an das ServiceAccount
- Dedizierte PodSecurityPolicy (für Kubernetes 1.25 und früher)
- Der Bediener selbst

-

Der Cluster-scoped Trident Operator verwaltet Ressourcen, die mit einer Astra Trident Installation auf Cluster-Ebene verbunden sind. Dies reduziert Fehler, die bei der Verwaltung von Clusterressourcen mit einem Namespace-Scoped-Operator auftreten können. Dies ist wichtig für die Selbstheilung und das Patching.

** Verheilen cappeckelT **

Der Bediener überwacht die Installation von Astra Trident und ergreift aktiv Maßnahmen, um Probleme wie das Löschen der Implementierung oder das versehentliche Ändern der Implementierung zu beheben. A `trident-operator-<generated-id>` Pod wird erstellt, der A zugeordnet `TridentOrchestrator` CR mit einer Astra Trident Installation. Dadurch wird sichergestellt, dass nur eine Instanz von Astra Trident im Cluster vorhanden ist und das Setup kontrolliert, um sicherzustellen, dass die Installation idempotent ist. Wenn Änderungen an der Installation vorgenommen werden (z. B. Löschen der Bereitstellung oder Knotendemonstrations), identifiziert der Bediener diese und korrigiert sie einzeln.

 Vermittlhat Updates für vorhandene InstalleIT

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.

 Retinstallbeam

Der im Cluster enthaltene Trident Operator ermöglicht die saubere Entfernung von im Cluster-Umfang enthaltenen Ressourcen. Benutzer können Astra Trident vollständig deinstallieren und einfach neu installieren.

 für Kubernetes Upgrade

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.

 Cluster Management unter Verwendung von BlueXP (früher 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.

Wird Verwendet `tridentctl`

Wenn Sie bereits eine Implementierung haben, die ein Upgrade durchgeführt werden muss oder eine stark angepasste Implementierung benötigen, sollten Sie dies in Betracht ziehen . Dies ist die herkömmliche Methode der Implementierung von Astra Trident.

Das können Sie 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.



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.

Wählen Sie die entsprechende `Installation_method_` aus.

Wählen Sie den Installationsmodus aus

Bestimmen Sie Ihren Bereitstellungsprozess auf der Grundlage des von Ihrem Unternehmen benötigten *Installations-Modus* (Standard, Offline oder Remote).

Standardinstallation

Dies ist der einfachste Weg, Astra Trident zu installieren und funktioniert für die meisten Umgebungen, die keine Netzwerkeinschränkungen auferlegen. Im Standardinstallationsmodus werden standardmäßig erforderliche Trident-Datenbanken verwendet (`docker.io`) Und CSI (`registry.k8s.io`) Bilder.

Wenn Sie den Standardmodus verwenden, können Sie das Astra Trident-Installationsprogramm:

- Ruft die Container-Images über das Internet ab
- Erstellt eine Implementierung oder Node-Demonset, bei dem Astra Trident Pods auf allen teilnahmeberechtigten Nodes im Kubernetes Cluster gespinnt werden

Offline-Installation

Der Offline-Installationsmodus kann an einem luftgekapselten oder sicheren Ort erforderlich sein. In diesem Szenario können Sie eine einzelne private, gespiegelte Registry oder zwei gespiegelte Registryrien erstellen, um die erforderlichen Trident- und CSI-Images zu speichern.



Unabhängig von Ihrer Registrierungskonfiguration müssen CSI-Bilder in einer Registrierung enthalten sein.

Remote-Installation

Hier finden Sie einen allgemeinen Überblick über den Remote-Installationsprozess:

- 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.

Wählen Sie den Prozess basierend auf Methode und Modus aus

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

| Method | Installationsmodus |
|------------------------------|--|
| Trident-Operator (manuell) | "Standardinstallation" "Offline-Installation" |
| Betreiber von Trident (Helm) | "Standardinstallation" "Offline-Installation" |
| tridentctl | "Standard- oder Offline-Installation" |

Wechseln zwischen den Installationsmethoden

Sie können sich entscheiden, Ihre Installationsmethode zu ändern. Bevor Sie dies tun, sollten Sie folgendes bedenken:

- Verwenden Sie immer die gleiche Methode für die Installation und Deinstallation von Astra Trident. 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 stattdessen 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.

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.

Installation über den Trident Operator

Manuelle Implementierung des Trident-Mitarbeiters (Standard-Modus)

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

Kritische Informationen zu Astra Trident 23.07

Sie müssen die folgenden wichtigen Informationen über Astra Trident lesen.

Informationen über Astra TriperelT

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

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

Trident-Operator kann manuell implementiert und Trident installiert werden

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

Bevor Sie beginnen

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



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

1. Überprüfen Sie Ihre Kubernetes Version:

```
kubectl version
```

2. Überprüfung der Berechtigungen für Cluster-Administratoren:

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

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

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

Schritt 1: Laden Sie das Trident Installer-Paket herunter

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

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

Schritt 2: Erstellen Sie die TridentOrchestrator CRD.

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

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

Schritt 3: Implementieren Sie den Trident-Operator

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

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

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

Bevor Sie beginnen

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

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

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

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

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

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

Schritte

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

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

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

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



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

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

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

```

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

kubect1 describe torc trident

Name:          trident
Namespace:
Labels:        <none>
Annotations:   <none>
API Version:   trident.netapp.io/v1
Kind:          TridentOrchestrator
...
Spec:
  Debug:       true
  Namespace:   trident
Status:
  Current Installation Params:
    IPv6:              false
    Autosupport Hostname:
    Autosupport Image:      netapp/trident-autosupport:23.07
    Autosupport Proxy:
    Autosupport Serial Number:
    Debug:              true
    Image Pull Secrets:
    Image Registry:
    k8sTimeout:         30
    Kubelet Dir:        /var/lib/kubelet
    Log Format:          text
    Silence Autosupport: false
    Trident Image:      netapp/trident:23.07.1
  Message:            Trident installed Namespace:
trident
  Status:              Installed
  Version:             v23.07.1
Events:
  Type Reason Age From Message ---- -
-----
Normal
Installing 74s trident-operator.netapp.io Installing Trident Normal
Installed 67s trident-operator.netapp.io Trident installed

```

Überprüfen Sie die Installation

Die Installation kann auf verschiedene Weise überprüft werden.

Wird Verwendet `TridentOrchestrator` Status

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

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

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

Den Status der Pod-Erstellung verwenden

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

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

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

Wird Verwendet `tridentctl`

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

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

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

Manuelles Bereitstellen des Trident-Mitarbeiters (Offline-Modus)

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

Kritische Informationen zu Astra Trident 23.07

Sie müssen die folgenden wichtigen Informationen über Astra Trident lesen.

** Informationen über Astra TriperelT **

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

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

Trident-Operator kann manuell implementiert und Trident installiert werden

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

Bevor Sie beginnen

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



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

1. Überprüfen Sie Ihre Kubernetes Version:

```
kubectl version
```

2. Überprüfung der Berechtigungen für Cluster-Administratoren:

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

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

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

Schritt 1: Laden Sie das Trident Installer-Paket herunter

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

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

Schritt 2: Erstellen Sie die TridentOrchestrator CRD.

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

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

Schritt 3: Aktualisieren Sie den Registrierungsart im Operator

In `/deploy/operator.yaml`, Update `image: docker.io/netapp/trident-operator:23.07.1` So geben Sie den Speicherort Ihrer Bildregistrierung an. Ihr ["Trident und CSI-Images"](#) Kann in einer Registrierung oder in verschiedenen Registern gefunden werden, aber alle CSI-Images müssen sich in derselben Registrierung befinden. Beispiel:

- `image: <your-registry>/trident-operator:23.07.1` Wenn Ihre Bilder alle in einer Registrierung gespeichert sind.

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

Schritt 4: Implementieren des Trident-Operators

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

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

Bevor Sie beginnen

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

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

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

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

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

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

Schritte

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

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

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

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



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

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

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

Bilder in einer Registrierung

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

Bilder in verschiedenen Registern

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

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

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

Sie können jetzt die erstellen `TridentOrchestrator` Und Installation von Astra Trident durchführen. Optional können Sie weiter ["Anpassung der Trident Installation"](#) Verwenden der Attribute im `TridentOrchestrator` Spez. Das folgende Beispiel zeigt eine Installation, bei der sich Trident- und CSI-Bilder in verschiedenen Registern befinden.

```

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

kubect1 describe torc trident

Name:          trident
Namespace:
Labels:        <none>
Annotations:   <none>
API Version:   trident.netapp.io/v1
Kind:          TridentOrchestrator
...
Spec:
  Autosupport Image: <your-registry>/netapp/trident-autosupport:23.07
  Debug:              true
  Image Registry:    <your-registry>/sig-storage
  Namespace:         trident
  Trident Image:     <your-registry>/netapp/trident:23.07.1
Status:
  Current Installation Params:
    IPv6:              false
    Autosupport Hostname:
    Autosupport Image: <your-registry>/netapp/trident-
autosupport:23.07
    Autosupport Proxy:
    Autosupport Serial Number:
    Debug:              true
    Http Request Timeout: 90s
    Image Pull Secrets:
    Image Registry:    <your-registry>/sig-storage
    k8sTimeout:        30
    Kubelet Dir:       /var/lib/kubelet
    Log Format:         text
    Probe Port:        17546
    Silence Autosupport: false
    Trident Image:     <your-registry>/netapp/trident:23.07.1
  Message:            Trident installed
  Namespace:          trident
  Status:              Installed
  Version:             v23.07.1
Events:
  Type Reason Age From Message ---- -
-----
Normal
Installing 74s trident-operator.netapp.io Installing Trident Normal
Installed 67s trident-operator.netapp.io Trident installed

```

Überprüfen Sie die Installation

Die Installation kann auf verschiedene Weise überprüft werden.

Wird Verwendet `TridentOrchestrator` Status

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

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

Den Status der Pod-Erstellung verwenden

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

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

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

Wird Verwendet `tridentctl`

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

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

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

Trident Operator mit Helm (Standard-Modus) implementieren

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

Kritische Informationen zu Astra Trident 23.07.1

Sie müssen die folgenden wichtigen Informationen über Astra Trident lesen.

 Informationen über Astra TriperelT

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

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

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

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

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

Bevor Sie beginnen

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

Schritte

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

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

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

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



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

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

Konfigurationsdaten während der Installation übergeben

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

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

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

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

Konfigurationsoptionen

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

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

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

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

Allgemeines zu Controller-Pods und Node-Pods

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

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

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

Trident-Operator mit Helm (Offline-Modus) implementieren

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

Kritische Informationen zu Astra Trident 23.07

Sie müssen die folgenden wichtigen Informationen über Astra Trident lesen.

** Informationen über Astra TriperelT **

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

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

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

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

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

Bevor Sie beginnen

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

Schritte

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

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

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

Bilder in einer Registrierung

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

Bilder in verschiedenen Registern

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

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



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

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

Konfigurationsdaten während der Installation übergeben

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

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

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

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

Konfigurationsoptionen

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

| Option | Beschreibung | Standard |
|-------------------------------------|---|--------------|
| 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. | |
| 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 |

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

| Option | Beschreibung | Standard |
|---------------------------------------|--|---------------------|
| <code>tridentDisableAuditLog</code> | Deaktiviert den Astra Trident Audit-Logger. | <code>true</code> |
| <code>tridentLogLevel</code> | Ermöglicht die Festlegung der Protokollebene von Astra Trident auf: <code>trace</code> , <code>debug</code> , <code>info</code> , <code>warn</code> , <code>error</code> , Oder <code>fatal</code> . | <code>"info"</code> |
| <code>tridentDebug</code> | Ermöglicht das Festlegen der Protokollebene für Astra Trident <code>debug</code> . | <code>false</code> |
| <code>tridentLogWorkflows</code> | Ermöglicht die Aktivierung bestimmter Astra Trident Workflows für die Trace-Protokollierung oder Protokollunterdrückung. | <code>"</code> |
| <code>tridentLogLayers</code> | Ermöglicht die Aktivierung bestimmter Astra Trident-Ebenen für die Trace-Protokollierung oder Protokollunterdrückung. | <code>"</code> |
| <code>tridentImage</code> | Ermöglicht die vollständige Überschreibung des Images für Astra Trident. | <code>"</code> |
| <code>tridentImageTag</code> | Ermöglicht das Überschreiben des Tags des Images für Astra Trident. | <code>"</code> |
| <code>tridentProbePort</code> | Ermöglicht das Überschreiben des Standardports, der für Kubernetes Liveness/Readiness-Sonden verwendet wird. | <code>"</code> |
| <code>windows</code> | Ermöglicht die Installation von Astra Trident auf einem Windows Worker-Node. | <code>false</code> |
| <code>enableForceDetach</code> | Ermöglicht die Aktivierung der Funktion zum Abtrennen erzwingen. | <code>false</code> |
| <code>excludePodSecurityPolicy</code> | Schließt die Sicherheitsrichtlinie des Operator POD von der Erstellung aus. | <code>false</code> |

Wie es weiter geht

Anpassen der Trident Operator-Installation

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

Allgemeines zu Controller-Pods und Node-Pods

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

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

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

Konfigurationsoptionen



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

Diese Tabelle enthält Einzelheiten `TridentOrchestrator` Attribute.

| Parameter | Beschreibung | Standard |
|---------------------------------|---|------------------------------------|
| <code>namespace</code> | Namespace für die Installation von Astra Trident in | „Standard“ |
| <code>debug</code> | Aktivieren Sie das Debugging für Astra Trident | Falsch |
| <code>enableForceDetach</code> | <code>ontap-san</code> Und <code>ontap-san-economy</code> Nur. Arbeitet mit Kubernetes Non-Graceful Node Shutdown (NGNS), um Clusteradministratoren die Möglichkeit zu geben, Workloads mit gemounteten Volumes sicher auf neue Nodes zu migrieren, sollte ein Node fehlerhaft werden. Dies ist ein experimentelles Feature in 23.04. Siehe Details zum Ablösen von Krafteinwirkung Wichtige Hinweise. | <code>false</code> |
| <code>windows</code> | Einstellung auf <code>true</code> Ermöglicht die Installation auf Windows Worker-Knoten. | Falsch |
| <code>IPv6</code> | Installieren Sie Astra Trident über IPv6 | Falsch |
| <code>k8sTimeout</code> | Zeitüberschreitung für Kubernetes-Betrieb | 30 Sek. |
| <code>silenceAutosupport</code> | Senden Sie keine AutoSupport Bundles an NetApp Automatisch | Falsch |
| <code>autosupportImage</code> | Das Container-Image für AutoSupport Telemetrie | „netapp/Trident-AutoSupport:23.07“ |

| Parameter | Beschreibung | Standard |
|------------------------------|--|--|
| autosupportProxy | Die Adresse/den 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:23.07“ |
| imageRegistry | Pfad zur internen Registrierung des Formats <registry FQDN>[:port] [/subpath] | „K8s.gcr.io/SIG-Storage“ (K8s 1.19+) Oder „Quay.io/k8scsi“ |
| kubeletDir | Pfad zum kubelet-Verzeichnis auf dem Host | „/var/lib/kubelet“ |
| wipeout | Eine Liste der zu löschenden Ressourcen, um eine vollständige Entfernung von durchzuführen Astra Trident | |
| imagePullSecrets | Secrets, um Bilder aus einer internen Registrierung zu ziehen | |
| imagePullPolicy | Legt die BildPull-Richtlinie für den Trident-Operator fest. Gültige Werte sind: Always Um immer das Bild zu ziehen. IfNotPresent Nur wenn das Image nicht auf dem Node vorhanden ist, soll das Image kopiert werden. Never Nie das Bild ziehen. | IfNotPresent |
| controllerPluginNodeSelector | Zusätzliche Node-Auswahl für Pods Entspricht dem Format <code>pod.spec.nodeSelector</code> . | Kein Standard; optional |
| controllerPluginTolerations | Überschreibt Kubernetes-Toleranzen für Pods. Entspricht dem gleichen Format wie <code>pod.spec.Tolerations</code> . | Kein Standard; optional |
| nodePluginNodeSelector | Zusätzliche Node-Auswahl für Pods Entspricht dem Format <code>pod.spec.nodeSelector</code> . | Kein Standard; optional |
| nodePluginTolerations | Überschreibt Kubernetes-Toleranzen für Pods. Entspricht dem gleichen Format wie <code>pod.spec.Tolerations</code> . | Kein Standard; optional |



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

Details zum Ablösen von Kraffteinwirkung

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



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

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

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



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

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

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

Beispielkonfigurationen

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

Beispiel 1: Grundlegende benutzerdefinierte Konfiguration

Dies ist ein Beispiel für eine benutzerdefinierte Grundkonfiguration.

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

Beispiel 2: Implementierung mit Node-Auswahl

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

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

Beispiel 3: Bereitstellung auf Windows Worker-Nodes

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

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

Installieren Sie mit tridentctl

Installieren Sie mit tridentctl

Sie können Astra Trident mithilfe von installieren `tridentctl`. Dieser Prozess gilt für Installationen, bei denen die von Astra Trident benötigten Container-Images entweder in einer privaten Registrierung gespeichert werden oder nicht. Um Ihre anzupassen `tridentctl` Die Bereitstellung finden Sie unter "[Tridentctl-Implementierung anpassen](#)".

Kritische Informationen zu Astra Trident 23.07

Sie müssen die folgenden wichtigen Informationen über Astra Trident lesen.

** Informationen über Astra TriperelT **

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

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

Installieren Sie Astra Trident mit `tridentctl`

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 Installationspaket von Astra Trident erstellt einen Trident Pod, konfiguriert die CRD-Objekte, die zur Aufrechterhaltung des Zustands verwendet werden, und initialisiert die CSI-Sidecars, um Aktionen wie die Bereitstellung und das Anschließen von Volumes an Cluster-Hosts durchzuführen. Laden Sie die neueste Version des Trident Installationsprogramms herunter und extrahieren Sie sie aus "[Die Sektion Assets auf GitHub](#)". Aktualisieren Sie `<trident-installer-XX.XX.X.tar.gz>` im Beispiel mit Ihrer ausgewählten Astra Trident Version.

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

Schritt: Installieren Sie Astra Trident

Installieren Sie Astra Trident im gewünschten Namespace, indem Sie den ausführen `tridentctl install` Befehl. Sie können weitere Argumente hinzufügen, um den Speicherort der Bildregistrierung anzugeben.

Standardmodus

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

Bilder in einer Registrierung

```
./tridentctl install -n trident --image-registry <your-registry>  
--autosupport-image <your-registry>/trident-autosupport:23.07 --trident  
-image <your-registry>/trident:23.07.1
```

Bilder in verschiedenen Registern

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

```
./tridentctl install -n trident --image-registry <your-registry>/sig-  
storage --autosupport-image <your-registry>/netapp/trident-  
autosupport:23.07 --trident-image <your-  
registry>/netapp/trident:23.07.1
```

Ihr Installationsstatus sollte so aussehen.

```
....  
INFO Starting Trident installation.                namespace=trident  
INFO Created service account.  
INFO Created cluster role.  
INFO Created cluster role binding.  
INFO Added finalizers to custom resource definitions.  
INFO Created Trident service.  
INFO Created Trident secret.  
INFO Created Trident deployment.  
INFO Created Trident daemonset.  
INFO Waiting for Trident pod to start.  
INFO Trident pod started.                          namespace=trident  
pod=trident-controller-679648bd45-cv2mx  
INFO Waiting for Trident REST interface.  
INFO Trident REST interface is up.                version=23.07.1  
INFO Trident installation succeeded.  
....
```

Überprüfen Sie die Installation

Sie können Ihre Installation mithilfe des POD-Erstellungsstatus oder überprüfen tridentctl.

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-679648bd45-cv2mx | 6/6 | Running | 0 | 5m29s |
| trident-node-linux-vgc8n | 2/2 | Running | 0 | 5m29s |



Wenn das Installationsprogramm nicht erfolgreich abgeschlossen wurde, oder `trident-controller-<generated id>` (`trident-csi-<generated id>` In Versionen vor 23.01) hat keinen **laufenden** Status, die Plattform wurde nicht installiert. Nutzung `-d` Bis "[Aktivieren Sie den Debug-Modus](#)" Und das Problem beheben.

Wird Verwendet `tridentctl`

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

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

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

Beispielkonfigurationen

Beispiel 1: Ausführung von Astra Trident auf Windows Nodes aktivieren

So aktivieren Sie die Ausführung von Astra Trident auf Windows Nodes:

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

Beispiel 2: Force Abtrennen aktivieren

Weitere Informationen zum gewaltsam Lösen finden Sie unter "[Anpassen der Trident Operator-Installation](#)".

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

Die tridentctl-Installation anpassen

Mit dem Astra Trident Installer können Sie die Installation 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 `tridentctl` 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 `tridentctl` 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 `tridentctl` angeben `--kubelet-dir`.

Wenn Sie die Installation anpassen müssen, die über die Argumente des Installers hinausgeht, können Sie auch die Bereitstellungsdateien anpassen. Verwenden der `tridentctl` Parameter `--generate-custom-yaml` Der Parameter erstellt die folgenden YAML-Dateien im Installationsprogramm `tridentctl` 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 `tridentctl` Parameter `--use-custom-yaml` Um Ihre benutzerdefinierte Bereitstellung zu installieren.

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

Nutzen Sie Astra Trident

Bereiten Sie den Knoten „Worker“ vor

Alle Worker-Nodes im Kubernetes-Cluster müssen in der Lage sein, die Volumes, die Sie für Ihre Pods bereitgestellt haben, zu mounten. Um die Worker-Knoten vorzubereiten, müssen Sie die NFS- oder iSCSI-Tools auf der Grundlage Ihrer Treiberauswahl installieren.

Auswahl der richtigen Werkzeuge

Wenn Sie eine Kombination von Treibern verwenden, sollten Sie NFS- und iSCSI-Tools installieren.

NFS Tools

Installieren Sie die NFS Tools, sofern Sie Folgendes verwenden: `ontap-nas`, `ontap-nas-economy`, `ontap-nas-flexgroup`, `azure-netapp-files`, `gcp-cvs`

iSCSI-Tools

Installieren Sie die iSCSI-Tools, wenn Sie Folgendes verwenden: `ontap-san`, `ontap-san-economy`, `solidfire-san`



In aktuellen Versionen von RedHat CoreOS sind standardmäßig NFS und iSCSI installiert.

Ermittlung des Node-Service

Astra Trident versucht automatisch zu erkennen, ob der Node iSCSI- oder NFS-Services ausführen kann.



Die Ermittlung des Node-Service erkennt erkannte Services, gewährleistet jedoch nicht, dass Services ordnungsgemäß konfiguriert wurden. Umgekehrt kann das Fehlen eines entdeckten Service nicht garantieren, dass die Volume-Bereitstellung fehlschlägt.

Überprüfen Sie Ereignisse

Astra Trident erstellt Ereignisse für den Node zur Identifizierung der erkannten Services. Um diese Ereignisse zu überprüfen, führen Sie folgende Schritte aus:

```
kubectl get event -A --field-selector involvedObject.name=<Kubernetes node name>
```

Überprüfen Sie erkannte Services

Astra Trident erkennt aktivierte Services für jeden Knoten auf der Trident Node CR. Um die ermittelten Dienste anzuzeigen, führen Sie folgende Schritte aus:

```
tridentctl get node -o wide -n <Trident namespace>
```


NFS Volumes

Installieren Sie die NFS-Tools unter Verwendung der Befehle für Ihr Betriebssystem. Stellen Sie sicher, dass der NFS-Dienst während des Bootens gestartet wird.

RHEL 8 ODER HÖHER

```
sudo yum install -y nfs-utils
```

Ubuntu

```
sudo apt-get install -y nfs-common
```



Starten Sie die Worker-Nodes nach der Installation der NFS-Tools neu, um einen Fehler beim Anschließen von Volumes an Container zu vermeiden.

ISCSI-Volumes

Astra Trident kann automatisch eine iSCSI-Sitzung einrichten, LUNs scannen und Multipath-Geräte erkennen, sie formatieren und auf einem Pod mounten.

ISCSI-Funktionen zur Selbstreparatur

Bei ONTAP Systemen führt Astra Trident alle fünf Minuten iSCSI-Selbsteilung aus und bietet folgende Vorteile:

1. * Identifizieren Sie den gewünschten iSCSI-Sitzungsstatus und den aktuellen iSCSI-Sitzungsstatus.
2. **Vergleichen** der gewünschte Zustand mit dem aktuellen Zustand, um notwendige Reparaturen zu identifizieren. Astra Trident ermittelt Reparaturprioritäten und wann Maßnahmen ergriffen werden müssen.
3. **Durchführung von Reparaturen** erforderlich, um den aktuellen iSCSI-Sitzungsstatus auf den gewünschten iSCSI-Sitzungsstatus zurückzusetzen.



Protokolle der Selbstheilungsaktivität befinden sich im `trident-main` Behälter auf dem jeweiligen Demonset Pod. Um Protokolle anzuzeigen, müssen Sie festgelegt haben `debug` Auf „true“ bei der Installation von Astra Trident zu setzen.

Astra Trident iSCSI-Funktionen zur Selbstheilung verhindern:

- Veraltete oder ungesunde iSCSI-Sitzungen, die nach einem Problem mit der Netzwerkverbindung auftreten können. Im Falle einer veralteten Sitzung wartet Astra Trident sieben Minuten vor der Anmeldung, um die Verbindung mit einem Portal wiederherzustellen.



Wenn beispielsweise CHAP-Schlüssel auf dem Speicher-Controller gedreht wurden und die Verbindung zum Netzwerk unterbrochen wird, können die alten (*Inated*) CHAP-Schlüssel bestehen bleiben. Selbstheilung kann dies erkennen und die Sitzung automatisch wiederherstellen, um die aktualisierten CHAP-Schlüssel anzuwenden.

- iSCSI-Sitzungen fehlen
- LUNs sind nicht vorhanden

Installieren Sie die iSCSI-Tools

Installieren Sie die iSCSI-Tools mit den Befehlen für Ihr Betriebssystem.

Bevor Sie beginnen

- Jeder Node im Kubernetes-Cluster muss über einen eindeutigen IQN verfügen. **Dies ist eine notwendige Voraussetzung.**
- Bei Verwendung von RHCOS Version 4.5 oder höher oder einer anderen RHEL-kompatiblen Linux-Distribution mit dem `solidfire-san` Treiber und Element OS 12.5 oder früher: Stellen Sie sicher, dass der CHAP-Authentifizierungsalgorithmus auf MD5 in eingestellt ist `/etc/iscsi/iscsid.conf`. Sichere, FIPS-konforme CHAP-Algorithmen SHA1, SHA-256 und SHA3-256 sind mit Element 12.7 erhältlich.

```
sudo sed -i 's/^\(node.session.auth.chap_algs\) .*/\1 = MD5/'  
/etc/iscsi/iscsid.conf
```

- Geben Sie bei Verwendung von Worker-Nodes, die RHEL/RedHat CoreOS mit iSCSI PVS ausführen, die an `discard` MountOption in StorageClass für die Inline-Speicherplatzrückgewinnung. Siehe "[Red hat-Dokumentation](#)".

RHEL 8 ODER HÖHER

1. Installieren Sie die folgenden Systempakete:

```
sudo yum install -y lsscsi iscsi-initiator-utils sg3_utils device-  
mapper-multipath
```

2. Überprüfen Sie, ob die Version von iscsi-Initiator-utils 6.2.0.874-2.el7 oder höher ist:

```
rpm -q iscsi-initiator-utils
```

3. Scannen auf manuell einstellen:

```
sudo sed -i 's/^\(node.session.scan\) .*/\1 = manual/'  
/etc/iscsi/iscsid.conf
```

4. Multipathing aktivieren:

```
sudo mpathconf --enable --with_multipathd y --find_multipaths n
```



Unbedingt `etc/multipath.conf` Enthält `find_multipaths no` Unter `defaults`.

5. Stellen Sie das sicher `iscsid` Und `multipathd` Laufen:

```
sudo systemctl enable --now iscsid multipathd
```

6. Aktivieren und starten `iscsi`:

```
sudo systemctl enable --now iscsi
```

Ubuntu

1. Installieren Sie die folgenden Systempakete:

```
sudo apt-get install -y open-iscsi lsscsi sg3-utils multipath-tools  
scsitools
```

2. Stellen Sie sicher, dass Open-iscsi-Version 2.0.874-5ubuntu2.10 oder höher (für bionic) oder 2.0.874-7.1ubuntu6.1 oder höher (für Brennweite) ist:

```
dpkg -l open-iscsi
```

3. Scannen auf manuell einstellen:

```
sudo sed -i 's/^\(node.session.scan\) .*/\1 = manual/'  
/etc/iscsi/iscsid.conf
```

4. Multipathing aktivieren:

```
sudo tee /etc/multipath.conf <<-'EOF'  
defaults {  
    user_friendly_names yes  
    find_multipaths no  
}  
EOF  
sudo systemctl enable --now multipath-tools.service  
sudo service multipath-tools restart
```



Unbedingt `etc/multipath.conf` Enthält `find_multipaths no` Unter `defaults`.

5. Stellen Sie das sicher `open-iscsi` Und `multipath-tools` Sind aktiviert und läuft:

```
sudo systemctl status multipath-tools  
sudo systemctl enable --now open-iscsi.service  
sudo systemctl status open-iscsi
```



Für Ubuntu 18.04, müssen Sie Ziel-Ports mit erkennen `iscsiadm` Vor dem Start `open-iscsi` Damit der iSCSI-Daemon gestartet werden kann. Alternativ können Sie den ändern `iscsi` Dienst zu starten `iscsid` Automatisch



Starten Sie die Worker-Knoten nach der Installation der iSCSI-Tools neu, um Fehler beim Anschließen von Volumes an Container zu vermeiden.

Konfiguration und Management von Back-Ends

Back-Ends konfigurieren

Ein Backend definiert die Beziehung zwischen Astra Trident und einem Storage-System. Er erzählt Astra Trident, wie man mit diesem Storage-System kommuniziert und wie Astra Trident Volumes darauf bereitstellen sollte.

Astra Trident stellt automatisch Storage-Pools aus Back-Ends bereit, die den von einer Storage-Klasse definierten Anforderungen entsprechen. Erfahren Sie, wie Sie das Backend für Ihr Storage-System konfigurieren.

- ["Konfigurieren Sie ein Azure NetApp Files-Backend"](#)
- ["Konfigurieren Sie ein Back-End für Cloud Volumes Service für Google Cloud Platform"](#)
- ["Konfigurieren Sie ein NetApp HCI- oder SolidFire-Backend"](#)
- ["Konfigurieren Sie ein Backend mit ONTAP- oder Cloud Volumes ONTAP-NAS-Treibern"](#)
- ["Konfigurieren Sie ein Backend mit ONTAP- oder Cloud Volumes ONTAP-SAN-Treibern"](#)
- ["Setzen Sie Astra Trident mit Amazon FSX für NetApp ONTAP ein"](#)

Azure NetApp Dateien

Konfigurieren Sie ein Azure NetApp Files-Backend

Sie können Azure NetApp Files als Backend für Astra Trident konfigurieren. Sie können NFS- und SMB-Volumes über ein Azure NetApp Files-Back-End einbinden.

Azure NetApp Files-Treiberdetails

Astra Trident bietet die folgenden Azure NetApp Files Storage-Treiber für die Kommunikation mit dem Cluster. Unterstützte Zugriffsmodi sind: *ReadWriteOnce* (RWO), *ReadOnlyMany* (ROX), *ReadWriteMany* (RWX), *ReadWriteOncePod* (RWOP).

| Treiber | Protokoll | VolumeModus | Unterstützte Zugriffsmodi | Unterstützte Filesysteme |
|--------------------|------------|-------------|---------------------------|--------------------------|
| azure-netapp-files | NFS SMB | Dateisystem | RWO, ROX, RWX, RWOP | nfs, smb |

Überlegungen

- Der Azure NetApp Files-Service unterstützt keine Volumes mit einer Größe von weniger als 100 GB. Astra Trident erstellt automatisch 100-GB-Volumes, wenn ein kleineres Volume benötigt wird.
- Astra Trident unterstützt SMB Volumes, die nur auf Windows Nodes laufenden Pods gemountet werden.

Konfiguration eines Azure NetApp Files-Backends wird vorbereitet

Bevor Sie Ihr Azure NetApp Files-Backend konfigurieren können, müssen Sie sicherstellen, dass die folgenden Anforderungen erfüllt sind.

Voraussetzungen für NFS und SMB Volumes



Wenn Sie Azure NetApp Files zum ersten Mal oder an einem neuen Standort verwenden, ist eine Erstkonfiguration erforderlich, um Azure NetApp Files einzurichten und ein NFS-Volume zu erstellen. Siehe ["Azure: Azure NetApp Files einrichten und ein NFS Volume erstellen"](#).

Um ein zu konfigurieren und zu verwenden ["Azure NetApp Dateien"](#) Back-End, Sie benötigen Folgendes:

- Ein Kapazitäts-Pool. Siehe ["Microsoft: Erstellen Sie einen Kapazitäts-Pool für Azure NetApp Files"](#).
- Ein an Azure NetApp Files delegiertes Subnetz. Siehe ["Microsoft: Delegieren Sie ein Subnetz an Azure NetApp Files"](#).
- subscriptionID Über ein Azure Abonnement mit aktiviertem Azure NetApp Files.
- tenantID, clientID, und clientSecret Von einem ["App-Registrierung"](#) In Azure Active Directory mit ausreichenden Berechtigungen für den Azure NetApp Files-Service. Die App-Registrierung sollte Folgendes verwenden:
 - Der Eigentümer oder die Rolle des Mitarbeiters ["Vordefiniert von Azure"](#).
 - A ["Benutzerdefinierte Beitragsrolle"](#) Auf Abonnementebene (assignableScopes) Mit den folgenden Berechtigungen, die auf nur das beschränkt sind, was Astra Trident erfordert. Nach dem Erstellen der benutzerdefinierten Rolle ["Weisen Sie die Rolle über das Azure-Portal zu"](#).

```
{
  "id": "/subscriptions/<subscription-id>/providers/Microsoft.Authorization/roleDefinitions/<role-definition-id>",
  "properties": {
    "roleName": "custom-role-with-limited-perms",
    "description": "custom role providing limited permissions",
    "assignableScopes": [
      "/subscriptions/<subscription-id>"
    ],
    "permissions": [
      {
        "actions": [
          "Microsoft.NetApp/netAppAccounts/capacityPools/read",
          "Microsoft.NetApp/netAppAccounts/capacityPools/write",
          "Microsoft.NetApp/netAppAccounts/capacityPools/volumes/read",
          "Microsoft.NetApp/netAppAccounts/capacityPools/volumes/write",
          "Microsoft.NetApp/netAppAccounts/capacityPools/volumes/delete",
          "Microsoft.NetApp/netAppAccounts/capacityPools/volumes/snapshots/read",
          "Microsoft.NetApp/netAppAccounts/capacityPools/volumes/snapshots/write",
          "Microsoft.NetApp/netAppAccounts/capacityPools/volumes/snapshots/delete"
        ]
      }
    ]
  }
}
```

```

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/MountTargets/read",
    "Microsoft.Network/virtualNetworks/read",
    "Microsoft.Network/virtualNetworks/subnets/read",

"Microsoft.Features/featureProviders/subscriptionFeatureRegistrations/read",

"Microsoft.Features/featureProviders/subscriptionFeatureRegistrations/write",

"Microsoft.Features/featureProviders/subscriptionFeatureRegistrations/delete",
    "Microsoft.Features/features/read",
    "Microsoft.Features/operations/read",
    "Microsoft.Features/providers/features/read",

"Microsoft.Features/providers/features/register/action",

"Microsoft.Features/providers/features/unregister/action",

"Microsoft.Features/subscriptionFeatureRegistrations/read"
    ],
    "notActions": [],
    "dataActions": [],
    "notDataActions": []
}
]
}
}

```

- Im Azure `location` Das enthält mindestens eine ["Delegiertes Subnetz"](#). Ab Trident 22.01 finden Sie das `location` Parameter ist ein erforderliches Feld auf der obersten Ebene der Backend-Konfigurationsdatei. In virtuellen Pools angegebene Standortwerte werden ignoriert.

Zusätzliche Anforderungen für SMB Volumes

Zur Erstellung eines SMB-Volumes müssen folgende Voraussetzungen erfüllt sein:

- Active Directory konfiguriert und mit Azure NetApp Files verbunden. Siehe ["Microsoft: Erstellen und Verwalten von Active Directory-Verbindungen für Azure NetApp Files"](#).
- Kubernetes-Cluster mit einem Linux-Controller-Knoten und mindestens einem Windows-Worker-Node, auf dem Windows Server 2019 ausgeführt wird. Astra Trident unterstützt SMB Volumes, die nur auf Windows Nodes laufenden Pods gemountet werden.
- Mindestens ein Astra Trident-Schlüssel mit Ihren Active Directory-Anmeldeinformationen, damit Azure NetApp Files sich bei Active Directory authentifizieren kann. Um Geheimnis zu erzeugen `smbcreds`:

```
kubectl create secret generic smbcreds --from-literal username=user
--from-literal password='password'
```

- Ein CSI-Proxy, der als Windows-Dienst konfiguriert ist. Zum Konfigurieren von A `csi-proxy` Weitere Informationen finden Sie unter "[GitHub: CSI-Proxy](#)" Oder "[GitHub: CSI Proxy für Windows](#)" Für Kubernetes-Knoten, die auf Windows ausgeführt werden.

Azure NetApp Files Back-End-Konfigurationsoptionen und -Beispiele

Informieren Sie sich über die Backend-Konfigurationsoptionen NFS und SMB für Azure NetApp Files und sehen Sie sich Konfigurationsbeispiele an.

Back-End-Konfigurationsoptionen

Astra Trident erstellt mithilfe Ihrer Backend-Konfiguration (Subnetz, virtuelles Netzwerk, Service-Level und Standort) Azure NetApp Files Volumes in Kapazitätspools, die am angeforderten Speicherort verfügbar sind und mit dem angeforderten Service-Level und Subnetz übereinstimmen.



Astra Trident unterstützt keine manuellen QoS-Kapazitäts-Pools.

Azure NetApp Files Back-Ends bieten diese Konfigurationsoptionen.

| Parameter | Beschreibung | Standard |
|-------------------|---|---------------------------------------|
| version | | Immer 1 |
| storageDriverName | Name des Speichertreibers | „azure-netapp-Files“ |
| backendName | Benutzerdefinierter Name oder das Storage-Backend | Treibername + „_“ + zufällige Zeichen |
| subscriptionID | Die Abonnement-ID Ihres Azure Abonnements | |
| tenantID | Die Mandanten-ID aus einer App-Registrierung | |
| clientID | Die Client-ID aus einer App-Registrierung | |
| clientSecret | Der Client-Schlüssel aus einer App-Registrierung | |
| serviceLevel | Einer von Standard, Premium, Oder Ultra | „“ (zufällig) |
| location | Name des Azure Speicherorts, an dem die neuen Volumes erstellt werden | |
| resourceGroups | Liste der Ressourcengruppen zum Filtern ermittelter Ressourcen | „[]“ (kein Filter) |
| netappAccounts | Liste von NetApp Accounts zur Filterung erkannter Ressourcen | „[]“ (kein Filter) |

| Parameter | Beschreibung | Standard |
|-----------------|--|--------------------------------------|
| capacityPools | Liste der Kapazitäts-Pools zur Filterung erkannter Ressourcen | „[]“ (kein Filter, zufällig) |
| virtualNetwork | Name eines virtuellen Netzwerks mit einem delegierten Subnetz | “ |
| subnet | Name eines an delegierten Subnetzes Microsoft.Netapp/volumes | “ |
| networkFeatures | Eventuell Set von vnet-Funktionen für ein Volumen Basic Oder Standard. Netzwerkfunktionen sind nicht in allen Regionen verfügbar und müssen möglicherweise in einem Abonnement aktiviert werden. Angabe networkFeatures Wenn die Funktion nicht aktiviert ist, schlägt die Volume-Bereitstellung fehl. | “ |
| nfsMountOptions | Engmaschige Kontrolle der NFS-Mount-Optionen Für SMB Volumes ignoriert. Um Volumes mit NFS-Version 4.1 einzubinden, beinhalten nfsvers=4 Wählen Sie in der Liste mit durch Komma getrennten Mount-Optionen NFS v4.1 aus. Mount-Optionen, die in einer Storage-Klassen-Definition festgelegt sind, überschreiben Mount-Optionen, die in der Backend-Konfiguration festgelegt sind. | „Nfsvers=3“ |
| limitVolumeSize | Bereitstellung fehlgeschlagen, wenn die angeforderte Volume-Größe über diesem Wert liegt | “ (nicht standardmäßig durchgesetzt) |
| debugTraceFlags | Fehler-Flags bei der Fehlerbehebung beheben. Beispiel: \{"api": false, "method": true, "discovery": true}. Verwenden Sie dies nur, wenn Sie Fehler beheben und einen detaillierten Log Dump benötigen. | Null |

| Parameter | Beschreibung | Standard |
|-----------|---|------------------|
| nasType | Konfiguration der Erstellung von NFS- oder SMB-Volumes Die Optionen lauten <code>nfs</code> , <code>smb</code> Oder <code>null</code> . Einstellung auf <code>null</code> setzt standardmäßig auf NFS-Volumes. | <code>nfs</code> |



Weitere Informationen zu den Netzwerkfunktionen finden Sie unter ["Konfigurieren Sie Netzwerkfunktionen für ein Azure NetApp Files Volume"](#).

Erforderliche Berechtigungen und Ressourcen

Wenn Sie beim Erstellen einer PVC den Fehler „Keine Kapazitätspools gefunden“ erhalten, sind Ihre App-Registrierung wahrscheinlich nicht über die erforderlichen Berechtigungen und Ressourcen (Subnetz, virtuelles Netzwerk, Kapazitäts-Pool) verbunden. Wenn Debug aktiviert ist, protokolliert Astra Trident die Azure Ressourcen, die bei der Erstellung des Backend ermittelt wurden. Überprüfen Sie, ob eine geeignete Rolle verwendet wird.

Die Werte für `resourceGroups`, `netappAccounts`, `capacityPools`, `virtualNetwork`, und `subnet` Kann mit kurzen oder vollqualifizierten Namen angegeben werden. In den meisten Fällen werden vollqualifizierte Namen empfohlen, da kurze Namen mehrere Ressourcen mit demselben Namen entsprechen können.

Der `resourceGroups`, `netappAccounts`, und `capacityPools` Werte sind Filter, die die ermittelten Ressourcen auf die in diesem Storage-Back-End verfügbaren Personen beschränken und in beliebiger Kombination angegeben werden können. Vollqualifizierte Namen folgen diesem Format:

| Typ | Formatieren |
|---------------------|---|
| Ressourcengruppe | <code><Ressourcengruppe></code> |
| NetApp Konto | <code><Resource Group>/<netapp Account></code> |
| Kapazitäts-Pool | <code><Resource Group>/<netapp Account>/<Capacity Pool></code> |
| Virtuelles Netzwerk | <code><Ressourcengruppe>/<virtuelles Netzwerk></code> |
| Subnetz | <code><Ressourcengruppe>/<virtuelles Netzwerk>/<Subnetz></code> |

Volume-Provisionierung

Sie können die standardmäßige Volume-Bereitstellung steuern, indem Sie die folgenden Optionen in einem speziellen Abschnitt der Konfigurationsdatei angeben. Siehe [Beispielkonfigurationen](#) Entsprechende Details.

| Parameter | Beschreibung | Standard |
|-----------------|--|--|
| exportRule | Exportregeln für neue Volumes exportRule Muss eine kommagetrennte Liste beliebiger Kombinationen von IPv4-Adressen oder IPv4-Subnetzen in CIDR-Notation sein. Für SMB Volumes ignoriert. | „0.0.0.0/0“ |
| snapshotDir | Steuert die Sichtbarkeit des .Snapshot-Verzeichnisses | „Falsch“ |
| size | Die Standardgröße der neuen Volumes | „100 GB“ |
| unixPermissions | die unix-Berechtigungen neuer Volumes (4 Oktal-Ziffern). Für SMB Volumes ignoriert. | „“ (Vorschau-Funktion, erfordert Whitelisting im Abonnement) |

Beispielkonfigurationen

Beispiel 1: Minimale Konfiguration

Dies ist die absolute minimale Backend-Konfiguration. Mit dieser Konfiguration erkennt Astra Trident alle NetApp-Konten, Kapazitätspools und Subnetze, die an Azure NetApp Files am konfigurierten Standort delegiert wurden. Zudem werden neue Volumes zufällig in einem dieser Pools und Subnetze platziert. Weil `nasType` Wird weggelassen, das `nfs` Standard gilt und das Backend wird für NFS-Volumes bereitgestellt.

Diese Konfiguration ist ideal, wenn Sie gerade erst mit Azure NetApp Files beginnen und Dinge ausprobieren möchten, aber in der Praxis möchten Sie einen zusätzlichen Umfang für die bereitgestellten Volumes angeben.

```
---
version: 1
storageDriverName: azure-netapp-files
subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
tenantID: 68e4f836-edc1-fake-bff9-b2d865ee56cf
clientID: dd043f63-bf8e-fake-8076-8de91e5713aa
clientSecret: SECRET
location: eastus
```

Beispiel 2: Spezifische Service Level-Konfiguration mit Kapazitätspool-Filtern

Bei dieser Back-End-Konfiguration werden Volumes in Azure platziert `eastus` Standort in einem `Ultra` Kapazitäts-Pool: Astra Trident erkennt automatisch alle an Azure NetApp Files delegierten Subnetze an diesem Standort und platziert ein neues Volume zufällig in einem davon.

```
---
version: 1
storageDriverName: azure-netapp-files
subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
tenantID: 68e4f836-edc1-fake-bff9-b2d865ee56cf
clientID: dd043f63-bf8e-fake-8076-8de91e5713aa
clientSecret: SECRET
location: eastus
serviceLevel: Ultra
capacityPools:
- application-group-1/account-1/ultra-1
- application-group-1/account-1/ultra-2
```

Beispiel 3: Erweiterte Konfiguration

Diese Back-End-Konfiguration reduziert den Umfang der Volume-Platzierung auf ein einzelnes Subnetz und ändert auch einige Standardwerte für die Volume-Bereitstellung.

```
---
version: 1
storageDriverName: azure-netapp-files
subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
tenantID: 68e4f836-edc1-fake-bff9-b2d865ee56cf
clientID: dd043f63-bf8e-fake-8076-8de91e5713aa
clientSecret: SECRET
location: eastus
serviceLevel: Ultra
capacityPools:
- application-group-1/account-1/ultra-1
- application-group-1/account-1/ultra-2
virtualNetwork: my-virtual-network
subnet: my-subnet
networkFeatures: Standard
nfsMountOptions: vers=3,proto=tcp,timeo=600
limitVolumeSize: 500Gi
defaults:
  exportRule: 10.0.0.0/24,10.0.1.0/24,10.0.2.100
  snapshotDir: 'true'
  size: 200Gi
  unixPermissions: '0777'
```

Beispiel 4: Konfiguration des virtuellen Pools

Diese Back-End-Konfiguration definiert mehrere Storage-Pools in einer einzelnen Datei. Dies ist nützlich, wenn Sie über mehrere Kapazitäts-Pools verfügen, die unterschiedliche Service-Level unterstützen, und Sie Storage-Klassen in Kubernetes erstellen möchten, die diese unterstützen. Virtuelle Pool-Labels wurden verwendet, um die Pools basierend auf zu differenzieren *performance*.

```
---
version: 1
storageDriverName: azure-netapp-files
subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
tenantID: 68e4f836-edc1-fake-bff9-b2d865ee56cf
clientID: dd043f63-bf8e-fake-8076-8de91e5713aa
clientSecret: SECRET
location: eastus
resourceGroups:
- application-group-1
networkFeatures: Basic
nfsMountOptions: vers=3,proto=tcp,timeo=600
labels:
  cloud: azure
storage:
- labels:
  performance: gold
  serviceLevel: Ultra
  capacityPools:
  - ultra-1
  - ultra-2
  networkFeatures: Standard
- labels:
  performance: silver
  serviceLevel: Premium
  capacityPools:
  - premium-1
- labels:
  performance: bronze
  serviceLevel: Standard
  capacityPools:
  - standard-1
  - standard-2
```

Definitionen der Storage-Klassen

Im Folgenden `StorageClass` Definitionen beziehen sich auf die oben genannten Speicherpools.

Beispieldefinitionen mit `parameter.selector` Feld

Wird verwendet `parameter.selector` Sie können für jedes angeben `StorageClass` Der virtuelle Pool, der zum Hosten eines Volumes genutzt wird. Im Volume werden die Aspekte definiert, die im ausgewählten Pool definiert sind.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gold
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=gold"
allowVolumeExpansion: true
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: silver
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=silver"
allowVolumeExpansion: true
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: bronze
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=bronze"
allowVolumeExpansion: true
```

Beispieldefinitionen für SMB Volumes

Wird verwendet `nasType`, `node-stage-secret-name`, und `node-stage-secret-namespace`, Sie können ein SMB-Volume angeben und die erforderlichen Active Directory-Anmeldeinformationen angeben.

Beispiel 1: Grundlegende Konfiguration im Standard-Namespace

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: anf-sc-smb
provisioner: csi.trident.netapp.io
parameters:
  backendType: "azure-netapp-files"
  trident.netapp.io/nasType: "smb"
  csi.storage.k8s.io/node-stage-secret-name: "smbcreds"
  csi.storage.k8s.io/node-stage-secret-namespace: "default"
```

Beispiel 2: Unterschiedliche Geheimnisse pro Namespace verwenden

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: anf-sc-smb
provisioner: csi.trident.netapp.io
parameters:
  backendType: "azure-netapp-files"
  trident.netapp.io/nasType: "smb"
  csi.storage.k8s.io/node-stage-secret-name: "smbcreds"
  csi.storage.k8s.io/node-stage-secret-namespace: ${pvc.namespace}
```

Beispiel 3: Verschiedene Geheimnisse pro Volumen

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: anf-sc-smb
provisioner: csi.trident.netapp.io
parameters:
  backendType: "azure-netapp-files"
  trident.netapp.io/nasType: "smb"
  csi.storage.k8s.io/node-stage-secret-name: ${pvc.name}
  csi.storage.k8s.io/node-stage-secret-namespace: ${pvc.namespace}
```




nasType: `smb` Filter für Pools, die SMB-Volumes unterstützen nasType: `nfs` Oder
 nasType: `null` Filter für NFS Pools.

Erstellen Sie das Backend

Führen Sie nach dem Erstellen der Back-End-Konfigurationsdatei den folgenden Befehl aus:

```
tridentctl create backend -f <backend-file>
```

Wenn die Backend-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 logs
```

Nachdem Sie das Problem mit der Konfigurationsdatei identifiziert und korrigiert haben, können Sie den Befehl „Erstellen“ erneut ausführen.

Cloud Volumes Service für Google Cloud-Back-End konfigurieren

Erfahren Sie, wie Sie NetApp Cloud Volumes Service für Google Cloud mit den vorgegebenen Beispielkonfigurationen als Backend für Ihre Astra Trident Installation konfigurieren.

Treiberdetails zu Google Cloud

Astra Trident bietet die `gcp-cvs` Treiber für die Kommunikation mit dem Cluster. Unterstützte Zugriffsmodi sind: *ReadWriteOnce* (RWO), *ReadOnlyMany* (ROX), *ReadWriteMany* (RWX), *ReadWriteOncePod* (RWOP).

| Treiber | Protokoll | VolumeModu s | Unterstützte Zugriffsmodi | Unterstützte Filesysteme |
|----------------------|-----------|-----------------|---------------------------|--------------------------|
| <code>gcp-cvs</code> | NFS | Dateisystem | RWO, ROX, RWX, RWOP | <code>nfs</code> |

Erfahren Sie mehr über den Astra Trident Support für Cloud Volumes Service für Google Cloud

Astra Trident kann Cloud Volumes Service Volumes in einem von zwei erstellen "[Servicetypen](#)":

- **CVS-Performance:** Der Standard Astra Trident Service-Typ. Dieser Performance-optimierte Service-Typ ist ideal für Produktions-Workloads, die Performance schätzen. Der CVS-Performance-Servicetyp ist eine Hardwareoption, die Volumes mit einer Größe von mindestens 100 gib unterstützt. Sie können eine von auswählen "[Drei Service-Level](#)":
 - `standard`
 - `premium`
 - `extreme`
- **CVS:** Der CVS-Servicetyp bietet eine hohe zonale Verfügbarkeit bei begrenzten bis moderaten Leistungsstufen. Der CVS-Servicetyp ist eine Software-Option, die Storage Pools zur Unterstützung von

Volumes mit einer Größe von 1 gib verwendet. Der Speicherpool kann bis zu 50 Volumes enthalten, in denen sich alle Volumes die Kapazität und Performance des Pools teilen. Sie können eine von auswählen "[Zwei Service-Level](#)":

- `standardsw`
- `zoneredundantstandardsw`

Was Sie benötigen

Um den zu konfigurieren und zu verwenden "[Cloud Volumes Service für Google Cloud](#)" Back-End, Sie benötigen Folgendes:

- Ein Google Cloud Konto, das mit NetApp Cloud Volumes Service konfiguriert ist
- Projektnummer Ihres Google Cloud-Kontos
- Google Cloud-Servicekonto bei `netappcloudvolumes.admin` Rolle
- API-Schlüsseldatei für Ihr Cloud Volumes Service-Konto

Back-End-Konfigurationsoptionen

Jedes Back-End stellt Volumes in einer einzigen Google Cloud-Region bereit. Um Volumes in anderen Regionen zu erstellen, können Sie zusätzliche Back-Ends definieren.

| Parameter | Beschreibung | Standard |
|--------------------------------|--|---|
| <code>version</code> | | Immer 1 |
| <code>storageDriverName</code> | Name des Speichertreibers | „gcp-cvs“ |
| <code>backendName</code> | Benutzerdefinierter Name oder das Storage-Backend | Treibername + „_“ + Teil des API-Schlüssels |
| <code>storageClass</code> | Optionaler Parameter zur Angabe des CVS-Servicetyps. Nutzung <code>software</code> Wählen Sie den CVS-Diensttyp aus. Anderenfalls übernimmt Astra Trident den Servicetyp CVS-Performance (<code>hardware</code>). | |
| <code>storagePools</code> | CVS-Diensttyp nur. Optionaler Parameter zur Angabe von Speicherpools für die Volume-Erstellung. | |
| <code>projectNumber</code> | Google Cloud Account Projektnummer. Der Wert ist auf der Startseite des Google Cloud Portals zu finden. | |
| <code>hostProjectNumber</code> | Erforderlich bei Verwendung eines gemeinsamen VPC-Netzwerks. In diesem Szenario <code>projectNumber</code> ist das Service-Projekt, und <code>hostProjectNumber</code> ist das Hostprojekt. | |

| Parameter | Beschreibung | Standard |
|-----------------|--|--|
| apiRegion | <p>In der Google Cloud-Region, in der Astra Trident Cloud Volumes Service Volumes erstellt. Wenn regionenübergreifende Kubernetes-Cluster erstellt werden, werden Volumes in einem erstellt apiRegion Können in Workloads verwendet werden, die auf Nodes über mehrere Google Cloud Regionen hinweg geplant sind.</p> <p>Der regionale Verkehr verursacht zusätzliche Kosten.</p> | |
| apiKey | <p>API-Schlüssel für das Google Cloud-Dienstkonto bei netappcloudvolumes.admin Rolle:</p> <p>Er enthält den JSON-formatierten Inhalt der privaten Schlüsseldatei eines Google Cloud-Dienstkontos (wortgetreu in die Back-End-Konfigurationsdatei kopiert).</p> | |
| proxyURL | <p>Proxy-URL, wenn Proxyserver für die Verbindung mit dem CVS-Konto benötigt wird. Der Proxy-Server kann entweder ein HTTP-Proxy oder ein HTTPS-Proxy sein.</p> <p>Bei einem HTTPS-Proxy wird die Zertifikatvalidierung übersprungen, um die Verwendung von selbstsignierten Zertifikaten im Proxyserver zu ermöglichen.</p> <p>Proxy-Server mit aktivierter Authentifizierung werden nicht unterstützt.</p> | |
| nfsMountOptions | Engmaschige Kontrolle der NFS-Mount-Optionen | „Nfsvers=3“ |
| limitVolumeSize | Bereitstellung fehlgeschlagen, wenn die angeforderte Volume-Größe über diesem Wert liegt. | „ (nicht standardmäßig durchgesetzt) |
| serviceLevel | <p>Das CVS-Performance oder CVS Service-Level für neue Volumes.</p> <p>CVS-Performance Werte sind standard, premium, Oder extreme.</p> <p>CVS-Werte sind standardsw Oder zonedundantstandardsw.</p> | <p>CVS-Performance ist der Standard.</p> <p>Der CVS-Standardwert ist „standardsw“.</p> |
| network | Für Cloud Volumes Service Volumes verwendetes Google Cloud Netzwerk | „Standard“ |
| debugTraceFlags | <p>Fehler-Flags bei der Fehlerbehebung beheben. Beispiel: <code>\{"api":false, "method":true}</code>.</p> <p>Verwenden Sie dies nur, wenn Sie Fehler beheben und einen detaillierten Log Dump benötigen.</p> | Null |

| Parameter | Beschreibung | Standard |
|-------------------|---|----------|
| allowedTopologies | <p>Damit Sie regionsübergreifenden Zugriff ermöglichen, wird Ihre StorageClass-Definition für verwendet allowedTopologies Muss alle Regionen umfassen.</p> <p>Beispiel:</p> <ul style="list-style-type: none"> - key: topology.kubernetes.io/region values: - us-east1 - europe-west1 | |

Optionen zur Volume-Bereitstellung

Sie können die Standard-Volume-Bereitstellung im steuern defaults Abschnitt der Konfigurationsdatei.

| Parameter | Beschreibung | Standard |
|-----------------|---|---|
| exportRule | Die Exportregeln für neue Volumes. Muss eine kommasetrennte Liste beliebiger Kombinationen von IPv4-Adressen oder IPv4-Subnetzen in CIDR-Notation sein. | „0.0.0.0/0“ |
| snapshotDir | Zugriff auf die .snapshot Verzeichnis | „Falsch“ |
| snapshotReserve | Prozentsatz des für Snapshots reservierten Volumes | "" (CVS Standard 0 akzeptieren) |
| size | <p>Die Größe neuer Volumes.</p> <p>Die Mindestmenge von CVS-Performance beträgt 100 gib.</p> <p>CVS mindestens 1 gib.</p> | <p>Der Servicetyp CVS-Performance ist standardmäßig auf „100 gib“ eingestellt.</p> <p>CVS-Diensttyp setzt keine Standardeinstellung, erfordert jedoch mindestens 1 gib.</p> |

Beispiele für CVS-Performance-Diensttypen

Die folgenden Beispiele enthalten Beispielfiguren für den CVS-Performance-Servicetyp.

Beispiel 1: Minimale Konfiguration

Dies ist die minimale Backend-Konfiguration, die den standardmäßigen CVS-Performance-Servicetyp mit dem Standard-Service Level verwendet.

```
---
version: 1
storageDriverName: gcp-cvs
projectNumber: '012345678901'
apiRegion: us-west2
apiKey:
  type: service_account
  project_id: my-gcp-project
  private_key_id: "<id_value>"
  private_key: |
    -----BEGIN PRIVATE KEY-----
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m
    XsYg6gyxy4zq7OlwWgLwGa==
    -----END PRIVATE KEY-----
  client_email: cloudvolumes-admin-sa@my-gcp-
  project.iam.gserviceaccount.com
  client_id: '123456789012345678901'
```

```
auth_uri: https://accounts.google.com/o/oauth2/auth
token_uri: https://oauth2.googleapis.com/token
auth_provider_x509_cert_url:
https://www.googleapis.com/oauth2/v1/certs
client_x509_cert_url:
https://www.googleapis.com/robot/v1/metadata/x509/cloudvolumes-admin-
sa%40my-gcp-project.iam.gserviceaccount.com
```



```
auth_uri: https://accounts.google.com/o/oauth2/auth
token_uri: https://oauth2.googleapis.com/token
auth_provider_x509_cert_url:
https://www.googleapis.com/oauth2/v1/certs
client_x509_cert_url:
https://www.googleapis.com/robot/v1/metadata/x509/cloudvolumes-admin-
sa%40my-gcp-project.iam.gserviceaccount.com
proxyURL: http://proxy-server-hostname/
nfsMountOptions: vers=3,proto=tcp,timeo=600
limitVolumeSize: 10Ti
serviceLevel: premium
defaults:
  snapshotDir: 'true'
  snapshotReserve: '5'
  exportRule: 10.0.0.0/24,10.0.1.0/24,10.0.2.100
  size: 5Ti
```



```

znHczZsrtrtHisIsAbOguSaPIKeyAZNchRAGz1zZE4jK3bl/qp8B4Kws8zX5ojY9m
XsYg6gyxy4zq7OlwWgLwGa==
-----END PRIVATE KEY-----
client_email: cloudvolumes-admin-sa@my-gcp-
project.iam.gserviceaccount.com
client_id: '123456789012345678901'
auth_uri: https://accounts.google.com/o/oauth2/auth
token_uri: https://oauth2.googleapis.com/token
auth_provider_x509_cert_url:
https://www.googleapis.com/oauth2/v1/certs
client_x509_cert_url:
https://www.googleapis.com/robot/v1/metadata/x509/cloudvolumes-admin-
sa%40my-gcp-project.iam.gserviceaccount.com
nfsMountOptions: vers=3,proto=tcp,timeo=600
defaults:
  snapshotReserve: '5'
  exportRule: 0.0.0.0/0
labels:
  cloud: gcp
  region: us-west2
storage:
- labels:
  performance: extreme
  protection: extra
  serviceLevel: extreme
  defaults:
  snapshotDir: 'true'
  snapshotReserve: '10'
  exportRule: 10.0.0.0/24
- labels:
  performance: extreme
  protection: standard
  serviceLevel: extreme
- labels:
  performance: premium
  protection: extra
  serviceLevel: premium
  defaults:
  snapshotDir: 'true'
  snapshotReserve: '10'
- labels:
  performance: premium
  protection: standard
  serviceLevel: premium
- labels:
  performance: standard

```

```
serviceLevel: standard
```

Definitionen der Storage-Klassen

Die folgenden StorageClass-Definitionen gelten für das Beispiel der virtuellen Pool-Konfiguration. Wird verwendet `parameters.selector`, Sie können für jede StorageClass den virtuellen Pool angeben, der zum Hosten eines Volumes verwendet wird. Im Volume werden die Aspekte definiert, die im ausgewählten Pool definiert sind.

Beispiel für Storage-Klasse

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: cvs-extreme-extra-protection
provisioner: netapp.io/trident
parameters:
  selector: "performance=extreme; protection=extra"
allowVolumeExpansion: true
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: cvs-extreme-standard-protection
provisioner: netapp.io/trident
parameters:
  selector: "performance=premium; protection=standard"
allowVolumeExpansion: true
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: cvs-premium-extra-protection
provisioner: netapp.io/trident
parameters:
  selector: "performance=premium; protection=extra"
allowVolumeExpansion: true
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: cvs-premium
provisioner: netapp.io/trident
parameters:
  selector: "performance=premium; protection=standard"
allowVolumeExpansion: true
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: cvs-standard
provisioner: netapp.io/trident
parameters:
  selector: "performance=standard"
allowVolumeExpansion: true
```

```
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: cvs-extra-protection
provisioner: netapp.io/trident
parameters:
  selector: "protection=extra"
allowVolumeExpansion: true
```

- Die erste StorageClass (`cvs-extreme-extra-protection`) Karten zum ersten virtuellen Pool. Dies ist der einzige Pool, der eine extreme Performance mit einer Snapshot-Reserve von 10 % bietet.
- Die letzte StorageClass (`cvs-extra-protection`) Ruft alle Speicher-Pool, die eine Snapshot-Reserve von 10% bietet. Astra Trident entscheidet, welcher Virtual Pool ausgewählt wird und stellt sicher, dass die Anforderungen an die Snapshot-Reserve erfüllt werden.

Beispiele für CVS-Diensttypen

Die folgenden Beispiele enthalten Beispielkonfigurationen für den CVS-Servicetyp.


```
client_id: '123456789012345678901'  
auth_uri: https://accounts.google.com/o/oauth2/auth  
token_uri: https://oauth2.googleapis.com/token  
auth_provider_x509_cert_url:  
https://www.googleapis.com/oauth2/v1/certs  
client_x509_cert_url:  
https://www.googleapis.com/robot/v1/metadata/x509/cloudvolumes-admin-  
sa%40my-gcp-project.iam.gserviceaccount.com  
serviceLevel: standardsw
```

Beispiel 2: Konfiguration des Storage Pools

Diese Beispiel-Back-End-Konfiguration verwendet `storagePools`. So konfigurieren Sie einen Speicherpool:

```
---
version: 1
storageDriverName: gcp-cvs
backendName: gcp-std-so-with-pool
projectNumber: '531265380079'
apiRegion: europe-west1
apiKey:
  type: service_account
  project_id: cloud-native-data
  private_key_id: "<id_value>"
  private_key: |-
    -----BEGIN PRIVATE KEY-----
    MIIIEvAIBADANBgkqhkiG9w0BAQEFAASCbKwggSiAgEAAoIBAQDaT+Oui9FBAw19
    L1AGEkrYU5xd9K5NlO5jMkIFND5wCD+Nv+jd1GvtFRLaLK5RvXyF5wzvztmODNS+
    qtScpQ+5cFpQkuGtv9U9+N6qtuVYYO3b504Kp5CtqVPJCgMJaK2j8pZTIqUiMum/
    5/Y9oTbZrjAHSMsgJm2nHzFq2X0rqVmaHghI6ATm4DOuWx8XGWKGTGIPlc0qPqJlqS
    LLaWOH4VIZQZCAyW5IU9cAmwqHgdG0uhFNfCgMmED6PBUvVLsLvcq86X+QSWR9k
    ETqElj/sGCenPF7ti1DhGBFafd9hPnxg9PZY29ArEZwY9G/ZjZQX7WPgs0VvxiNR
    DxZRC3GXAgMBAAECggEACn5c59bG/qnVEVI1CwMAalM5M2z09JFh1L1ljKwntNPj
    Vilw2eTW2+UE7HbJru/S7KQgA5Dnn9kvCraEahPRuddUMrD0vG4kTl/IODV6uFuk
    Y0sZfbqd4jMUQ21smvGsqFzwloYWS5qzO1W83ivXH/HW/iqkmY2eW+EPRS/hwSSu
    SscR+SojI7PB0BWSJhlV4yqYf3vcd/D95el2CVHfRCkL85DKumeZ+yHEnpiXGZAE
    t8xSs4a500Pm6NHhevCw2a/UQ95/foXNUR450HtbjieJo5o+FF6EYZQGfU2ZHZO8
    37FBKuaJkdGW5xqaI9TL7aqkGkFMF4F2qvOZM+vy8QKBgQD4oVuOkJD1hkTHP86W
    esFlw1kpWyJR9ZA7LI0g/rVpslnX+XdDq0WQf4umdLNau5hYEH9LU6ZSGs1Xk3/B
    NHwR6OXFuqEKNiu83d0zSlHhTy7PZpOZdj5a/vVvQfPDMz7OvsqLRd7YCAbdzuQ0
    +Ahq0Ztwvg0HQ64hdW0ukpYRRwKBgQDgyHj98oqswoYuIa+pP1yS0pPwLmjwKyNm
    /HayzCp+Qjiiyy7Tzg8AUqlH1Ou83XbV428jvg7kDhO7PCCKFq+mMmfqHmTpb0Maq
    KpKnZg4ipsqPlyHNNNEoRmcailXbwIhCLewMqMrggUiLOmCw4PscL5nK+4GKu2XE1
    jLqjWAZFMQKBgFHkQ9XXRAJ1kR3XpGHOgn890pZOkCVSrqju6aUef/5KY1FCt8ew
    F/+aIxM2iQsvmWQYOvVCnhuY/F2GfAQ7d0om3decuwI0CX/xy7PjHmLXa2uaZs4
    WR17sLduj62RqXRLX0c0QkwBiNFyHbRcpdkZJQujbyMhBa+7j7SxT4BtAoGAWMWT
    UucocRXZm/pdvz9wteNH3YDwnJLMxm1KC06qMXbBoYrliY4sm3ywJWMC+iCd/H8A
    Gecxd/xVu5mA2L2N3KMq18Zhz8Th0G5DwKyDRJgOQ0Q46yuNXOoYEjlo4Wjyk8Me
    +tlQ8iK98E0UmZnhTgfSpSNElbz2AqnzQ3MN9uECgYAqdvvdVPnKGFvdtZ2DjyMoJ
    E89UIC41WjjJGmHsd8W65+3X0RwMzKMT6aZc5tK9J5dHvmWIETnbM+1TImdBFBfga
    NWOC6f3r2xbGXHhaWSl+nobpTuvlo56ZRJVvV7lFMsidzMuHH8pxfgNJemwA4P
    ThDHcejv035NNV6Kyo00tA==
    -----END PRIVATE KEY-----
  client_email: cloudvolumes-admin-sa@cloud-native-
  data.iam.gserviceaccount.com
```



```
client_id: '107071413297115343396'  
auth_uri: https://accounts.google.com/o/oauth2/auth  
token_uri: https://oauth2.googleapis.com/token  
auth_provider_x509_cert_url:  
https://www.googleapis.com/oauth2/v1/certs  
client_x509_cert_url:  
https://www.googleapis.com/robot/v1/metadata/x509/cloudvolumes-admin-  
sa%40cloud-native-data.iam.gserviceaccount.com  
storageClass: software  
zone: europe-west1-b  
network: default  
storagePools:  
- 1bc7f380-3314-6005-45e9-c7dc8c2d7509  
serviceLevel: Standardsw
```

Was kommt als Nächstes?

Führen Sie nach dem Erstellen der Back-End-Konfigurationsdatei den folgenden Befehl aus:

```
tridentctl create backend -f <backend-file>
```

Wenn die Backend-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 logs
```

Nachdem Sie das Problem mit der Konfigurationsdatei identifiziert und korrigiert haben, können Sie den Befehl „Erstellen“ erneut ausführen.

Konfigurieren Sie ein NetApp HCI- oder SolidFire-Backend

Erstellen und Verwenden eines Element Backend mit der Astra Trident Installation

Details zum Elementtreiber

Astra Trident bietet die `solidfire-san` Speichertreiber für die Kommunikation mit dem Cluster. Unterstützte Zugriffsmodi sind: *ReadWriteOnce* (RWO), *ReadOnly Many* (ROX), *ReadWriteMany* (RWX), *ReadWriteOncePod* (RWOP).

Der `solidfire-san` Der Speichertreiber unterstützt die Volume-Modi *File* und *Block*. Für das Filesystem VolumeMode erstellt Astra Trident ein Volume und erstellt ein Dateisystem. Der Dateisystem-Typ wird von StorageClass angegeben.

| Treiber | Protokoll | VolumeMode | Unterstützte Zugriffsmodi | Unterstützte Filesysteme |
|---------------|-----------|---------------|---------------------------|-------------------------------------|
| solidfire-san | ISCSI | Block-Storage | RWO, ROX, RWX, RWOP | Kein Dateisystem. Rohes Blockgerät. |
| solidfire-san | ISCSI | Dateisystem | RWO, RWOP | xfss, ext3, ext4 |

Bevor Sie beginnen

Sie benötigen Folgendes, bevor Sie ein Element-Backend erstellen.

- Ein unterstütztes Storage-System, auf dem die Element Software ausgeführt wird.
- Anmeldedaten für einen NetApp HCI/SolidFire Cluster-Administrator oder einen Mandantenbenutzer, der Volumes managen kann
- Alle Kubernetes-Worker-Nodes sollten die entsprechenden iSCSI-Tools installiert haben. Siehe ["Informationen zur Vorbereitung auf den Worker-Node"](#).

Back-End-Konfigurationsoptionen

Die Back-End-Konfigurationsoptionen finden Sie in der folgenden Tabelle:

| Parameter | Beschreibung | Standard |
|-------------------|--|--|
| version | | Immer 1 |
| storageDriverName | Name des Speichertreibers | Immer „solidfire-san“ |
| backendName | Benutzerdefinierter Name oder das Storage-Backend | IP-Adresse „SolidFire_“ + Storage (iSCSI) |
| Endpoint | MVIP für den SolidFire-Cluster mit Mandanten-Anmeldedaten | |
| SVIP | Speicher-IP-Adresse und -Port | |
| labels | Satz willkürlicher JSON-formatierter Etiketten für Volumes. | “ |
| TenantName | Zu verwendende Mandantenbezeichnung (wird erstellt, wenn sie nicht gefunden wurde) | |
| InitiatorIFace | Beschränken Sie den iSCSI-Datenverkehr auf eine bestimmte Host-Schnittstelle | „Standard“ |
| UseCHAP | Verwenden Sie CHAP zur Authentifizierung von iSCSI. Astra Trident verwendet CHAP. | Richtig |
| AccessGroups | Liste der zu verwendenden Zugriffsgruppen-IDs | Findet die ID einer Zugriffsgruppe namens „Dreizack“ |
| Types | QoS-Spezifikationen | |

| Parameter | Beschreibung | Standard |
|-----------------|--|--------------------------------------|
| limitVolumeSize | Bereitstellung fehlgeschlagen, wenn die angeforderte Volume-Größe über diesem Wert liegt | „ (nicht standardmäßig durchgesetzt) |
| debugTraceFlags | Fehler-Flags bei der Fehlerbehebung beheben. Beispiel: { „API“:false, „Methode“:true} | Null |



Verwenden Sie es nicht `debugTraceFlags` Es sei denn, Sie beheben Fehler und benötigen einen detaillierten Log Dump.

Beispiel 1: Back-End-Konfiguration für `solidfire-san` Treiber mit drei Lautstärketypen

Dieses Beispiel zeigt eine Backend-Datei mit CHAP-Authentifizierung und Modellierung von drei Volume-Typen mit spezifischen QoS-Garantien. Sehr wahrscheinlich würden Sie dann Storage-Klassen definieren, um jeden davon mit dem zu nutzen `IOPS` Parameter für Storage-Klasse.

```

---
version: 1
storageDriverName: solidfire-san
Endpoint: https://<user>:<password>@<mvip>/json-rpc/8.0
SVIP: "<svip>:3260"
TenantName: "<tenant>"
labels:
  k8scluster: dev1
  backend: dev1-element-cluster
UseCHAP: true
Types:
- Type: Bronze
  Qos:
    minIOPS: 1000
    maxIOPS: 2000
    burstIOPS: 4000
- Type: Silver
  Qos:
    minIOPS: 4000
    maxIOPS: 6000
    burstIOPS: 8000
- Type: Gold
  Qos:
    minIOPS: 6000
    maxIOPS: 8000
    burstIOPS: 10000

```

Beispiel 2: Back-End- und Storage-Class-Konfiguration für `solidfire-san` Treiber mit virtuellen Pools

Dieses Beispiel zeigt die mit virtuellen Pools zusammen mit StorageClasses konfigurierte Back-End-Definitionsdatei.

Astra Trident kopiert beim Provisioning die auf einem Storage-Pool vorhandenen Labels auf die Back-End-Storage-LUN. Storage-Administratoren können Labels je virtuellen Pool definieren und Volumes nach Label gruppieren.

In der unten gezeigten Beispiel-Backend-Definitionsdatei werden für alle Speicherpools spezifische Standardwerte festgelegt, die die definieren `type` Bei Silver. Die virtuellen Pools werden im definiert `storage` Abschnitt. In diesem Beispiel legen einige Speicherpools ihren eigenen Typ fest, und einige Pools überschreiben die oben festgelegten Standardwerte.

```
---
version: 1
storageDriverName: solidfire-san
Endpoint: https://<user>:<password>@<mvip>/json-rpc/8.0
SVIP: "<svip>:3260"
TenantName: "<tenant>"
UseCHAP: true
Types:
- Type: Bronze
  Qos:
    minIOPS: 1000
    maxIOPS: 2000
    burstIOPS: 4000
- Type: Silver
  Qos:
    minIOPS: 4000
    maxIOPS: 6000
    burstIOPS: 8000
- Type: Gold
  Qos:
    minIOPS: 6000
    maxIOPS: 8000
    burstIOPS: 10000
type: Silver
labels:
  store: solidfire
  k8scluster: dev-1-cluster
region: us-east-1
storage:
- labels:
  performance: gold
  cost: '4'
  zone: us-east-1a
  type: Gold
```

```
- labels:
  performance: silver
  cost: '3'
  zone: us-east-1b
  type: Silver
- labels:
  performance: bronze
  cost: '2'
  zone: us-east-1c
  type: Bronze
- labels:
  performance: silver
  cost: '1'
  zone: us-east-1d
```

Die folgenden StorageClass-Definitionen beziehen sich auf die oben genannten virtuellen Pools. Verwenden der `parameters.selector` Feld gibt in jeder StorageClass an, welche virtuellen Pools zum Hosten eines Volumes verwendet werden können. Auf dem Volume werden die Aspekte im ausgewählten virtuellen Pool definiert.

Die erste StorageClass (`solidfire-gold-four`) Wird dem ersten virtuellen Pool zugeordnet. Dies ist der einzige Pool, der Gold Performance mit einem bietet `Volume Type QoS` Von Gold. Die letzte StorageClass (`solidfire-silver`) Bezeichnet jeden Speicherpool, der eine silberne Leistung bietet. Astra Trident entscheidet, welcher virtuelle Pool ausgewählt wird und stellt sicher, dass die Storage-Anforderungen erfüllt werden.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: solidfire-gold-four
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=gold; cost=4"
  fsType: "ext4"
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: solidfire-silver-three
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=silver; cost=3"
  fsType: "ext4"
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: solidfire-bronze-two
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=bronze; cost=2"
  fsType: "ext4"
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: solidfire-silver-one
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=silver; cost=1"
  fsType: "ext4"
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: solidfire-silver
provisioner: csi.trident.netapp.io
parameters:
  selector: "performance=silver"
  fsType: "ext4"
```

Weitere Informationen

- ["Volume-Zugriffsgruppen"](#)

ONTAP SAN-Treiber

Übersicht über ONTAP SAN-Treiber

Erfahren Sie mehr über die Konfiguration eines ONTAP Backend mit ONTAP- und Cloud Volumes ONTAP-SAN-Treibern.

Details zum ONTAP-SAN-Treiber

Astra Trident bietet die folgenden SAN-Storage-Treiber für die Kommunikation mit dem ONTAP Cluster. Unterstützte Zugriffsmodi sind: *ReadWriteOnce* (RWO), *ReadOnly Many* (ROX), *ReadWriteMany* (RWX), *ReadWriteOncePod* (RWOP).



Wenn Sie Astra Control für Schutz, Recovery und Mobilität verwenden, lesen Sie bitte [Treiberkompatibilität bei Astra Control](#).

| Treiber | Protokoll | VolumeModus | Unterstützte Zugriffsmodi | Unterstützte Filesysteme |
|-------------------|-----------|---------------|---|------------------------------------|
| ontap-san | ISCSI | Block-Storage | RWO, ROX, RWX, RWOP | Kein Filesystem, rohes Block-Gerät |
| ontap-san | ISCSI | Dateisystem | RWO, RWOP ROX und RWX sind im Filesystem-Volume-Modus nicht verfügbar. | xf ^s , ext3, ext4 |
| ontap-san-economy | ISCSI | Block-Storage | RWO, ROX, RWX, RWOP | Kein Filesystem, rohes Block-Gerät |
| ontap-san-economy | ISCSI | Dateisystem | RWO, RWOP ROX und RWX sind im Filesystem-Volume-Modus nicht verfügbar. | xf ^s , ext3, ext4 |

Treiberkompatibilität bei Astra Control

Astra Control bietet nahtlosen Schutz, Disaster Recovery und Mobilität (Verschieben von Volumes zwischen Kubernetes Clustern) für Volumes, die mit der erstellt wurden `ontap-nas`, `ontap-nas-flexgroup`, und `ontap-san` Treiber. Siehe ["Voraussetzungen für die Astra Control Replikation"](#) Entsprechende Details.



- Nutzung `ontap-san-economy` Nur wenn die Nutzungszahl für persistente Volumes voraussichtlich höher ist als "[Unterstützte ONTAP-Volume-Größen](#)".
- Nutzung `ontap-nas-economy` Nur wenn die Nutzungszahl für persistente Volumes voraussichtlich höher ist als "[Unterstützte ONTAP-Volume-Größen](#)" Und das `ontap-san-economy` Treiber kann nicht verwendet werden.
- Verwenden Sie ihn nicht `ontap-nas-economy` Wenn Sie die Notwendigkeit von Datensicherung, Disaster Recovery oder Mobilität erwarten.

Benutzerberechtigungen

Astra Trident erwartet, dass er entweder als ONTAP- oder SVM-Administrator ausgeführt wird, in der Regel mit dem `admin` Cluster-Benutzer oder ein `vsadmin` SVM-Benutzer oder ein Benutzer mit einem anderen Namen und derselben Rolle. Astra Trident erwartet, dass bei Amazon FSX für Implementierungen von NetApp ONTAP, über das Cluster entweder als ONTAP- oder SVM-Administrator ausgeführt wird `fsxadmin` Benutzer oder A `vsadmin` SVM-Benutzer oder ein Benutzer mit einem anderen Namen und derselben Rolle. Der `fsxadmin` Der Benutzer ist ein eingeschränkter Ersatz für den Cluster-Admin-Benutzer.



Wenn Sie den verwenden `limitAggregateUsage` Parameter, Berechtigungen für Cluster-Admin sind erforderlich. Bei der Verwendung von Amazon FSX für NetApp ONTAP mit Astra Trident, das `limitAggregateUsage` Der Parameter funktioniert nicht mit dem `vsadmin` Und `fsxadmin` Benutzerkonten. Der Konfigurationsvorgang schlägt fehl, wenn Sie diesen Parameter angeben.

Es ist zwar möglich, eine restriktivere Rolle in ONTAP zu erstellen, die ein Trident-Treiber verwenden kann, wir empfehlen sie jedoch nicht. Bei den meisten neuen Versionen von Trident sind zusätzliche APIs erforderlich, die berücksichtigt werden müssten, was Upgrades schwierig und fehleranfällig macht.

Vorbereiten der Konfiguration des Back-End mit ONTAP-SAN-Treibern

Verstehen Sie die Anforderungen und Authentifizierungsoptionen für die Konfiguration eines ONTAP-Backends mit ONTAP-SAN-Treibern.

Anforderungen

Für alle ONTAP Back-Ends benötigt Astra Trident mindestens ein Aggregat, das der SVM zugewiesen ist.

Denken Sie daran, dass Sie auch mehr als einen Treiber ausführen können und Speicherklassen erstellen können, die auf den einen oder anderen verweisen. Beispielsweise könnten Sie A konfigurieren `san-dev` Klasse, die den verwendet `ontap-san` Fahrer und A `san-default` Klasse, die den verwendet `ontap-san-economy` Eins.

Alle Kubernetes-Worker-Nodes müssen über die entsprechenden iSCSI-Tools verfügen. Siehe "[Bereiten Sie den Knoten „Worker“ vor](#)" Entsprechende Details.

Authentifizieren Sie das ONTAP-Backend

Astra Trident bietet zwei Arten der Authentifizierung eines ONTAP-Backend.

- Anmeldeinformationsbasiert: Benutzername und Passwort für einen ONTAP-Benutzer mit den erforderlichen Berechtigungen. Es wird empfohlen, eine vordefinierte Sicherheits-Login-Rolle zu verwenden, wie z. B. `admin` Oder `vsadmin` Für maximale Kompatibilität mit ONTAP Versionen.

- Zertifikatsbasiert: Astra Trident kann auch mit einem ONTAP Cluster kommunizieren. Verwenden Sie dazu ein Zertifikat, das auf dem Backend installiert ist. Hier muss die Backend-Definition Base64-kodierte Werte des Client-Zertifikats, des Schlüssels und des vertrauenswürdigen CA-Zertifikats enthalten, sofern verwendet (empfohlen).

Sie können vorhandene Back-Ends aktualisieren, um zwischen auf Anmeldeinformationen basierenden und zertifikatbasierten Methoden zu verschieben. Es wird jedoch immer nur eine Authentifizierungsmethode unterstützt. Um zu einer anderen Authentifizierungsmethode zu wechseln, müssen Sie die vorhandene Methode von der Backend-Konfiguration entfernen.



Wenn Sie versuchen, **sowohl Anmeldeinformationen als auch Zertifikate** bereitzustellen, schlägt die Backend-Erstellung mit einem Fehler fehl, dass mehr als eine Authentifizierungsmethode in der Konfigurationsdatei angegeben wurde.

Aktivieren Sie die Anmeldeinformationsbasierte Authentifizierung

Astra Trident erfordert die Zugangsdaten für einen Administrator mit SVM-Umfang/Cluster-Umfang, um mit dem Backend von ONTAP zu kommunizieren. Es wird empfohlen, die Standard-vordefinierten Rollen wie zu verwenden `admin` Oder `vsadmin`. So ist gewährleistet, dass die Kompatibilität mit künftigen ONTAP Versionen gewährleistet ist, die FunktionsAPIs der künftigen Astra Trident Versionen bereitstellen können. Eine benutzerdefinierte Sicherheits-Login-Rolle kann mit Astra Trident erstellt und verwendet werden, wird aber nicht empfohlen.

Eine Beispiel-Back-End-Definition sieht folgendermaßen aus:

YAML

```
---
version: 1
backendName: ExampleBackend
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_nfs
username: vsadmin
password: password
```

JSON

```
{
  "version": 1,
  "backendName": "ExampleBackend",
  "storageDriverName": "ontap-san",
  "managementLIF": "10.0.0.1",
  "svm": "svm_nfs",
  "username": "vsadmin",
  "password": "password"
}
```

Beachten Sie, dass die Backend-Definition der einzige Ort ist, an dem die Anmeldeinformationen im reinen Text gespeichert werden. Nach der Erstellung des Backend werden Benutzernamen/Passwörter mit Base64 codiert und als Kubernetes Secrets gespeichert. Die Erstellung oder Aktualisierung eines Backend ist der einzige Schritt, der Kenntnisse über die Anmeldeinformationen erfordert. Daher ist dieser Vorgang nur für Administratoren und wird vom Kubernetes-/Storage-Administrator ausgeführt.

Aktivieren Sie die zertifikatbasierte Authentifizierung

Neue und vorhandene Back-Ends können ein Zertifikat verwenden und mit dem ONTAP-Back-End kommunizieren. In der Backend-Definition sind drei Parameter erforderlich.

- **ClientCertificate:** Base64-codierter Wert des Clientzertifikats.
- **ClientPrivateKey:** Base64-kodierte Wert des zugeordneten privaten Schlüssels.
- **Trusted CACertificate:** Base64-codierter Wert des vertrauenswürdigen CA-Zertifikats. Bei Verwendung einer vertrauenswürdigen CA muss dieser Parameter angegeben werden. Dies kann ignoriert werden, wenn keine vertrauenswürdige CA verwendet wird.

Ein typischer Workflow umfasst die folgenden Schritte.

Schritte

1. Erzeugen eines Clientzertifikats und eines Schlüssels. Legen Sie beim Generieren den allgemeinen Namen (CN) für den ONTAP-Benutzer fest, der sich authentifizieren soll als.

```
openssl req -x509 -nodes -days 1095 -newkey rsa:2048 -keyout k8senv.key  
-out k8senv.pem -subj "/C=US/ST=NC/L=RTP/O=NetApp/CN=admin"
```

2. Fügen Sie dem ONTAP-Cluster ein vertrauenswürdigen CA-Zertifikat hinzu. Dies kann möglicherweise bereits vom Storage-Administrator übernommen werden. Ignorieren, wenn keine vertrauenswürdige CA verwendet wird.

```
security certificate install -type server -cert-name <trusted-ca-cert-name> -vserver <vserver-name>  
ssl modify -vserver <vserver-name> -server-enabled true -client-enabled  
true -common-name <common-name> -serial <SN-from-trusted-CA-cert> -ca  
<cert-authority>
```

3. Installieren Sie das Client-Zertifikat und den Schlüssel (von Schritt 1) auf dem ONTAP-Cluster.

```
security certificate install -type client-ca -cert-name <certificate-name> -vserver <vserver-name>  
security ssl modify -vserver <vserver-name> -client-enabled true
```

4. Bestätigen Sie, dass die ONTAP-Sicherheitsanmeldungsrolle unterstützt wird `cert` Authentifizierungsmethode.

```
security login create -user-or-group-name admin -application ontapi
-authentication-method cert
security login create -user-or-group-name admin -application http
-authentication-method cert
```

5. Testen Sie die Authentifizierung mithilfe des generierten Zertifikats. <ONTAP Management LIF> und <vServer Name> durch Management-LIF-IP und SVM-Namen ersetzen.

```
curl -X POST -Lk https://<ONTAP-Management-
LIF>/servlets/netapp.servlets.admin.XMLrequest_filer --key k8senv.key
--cert ~/k8senv.pem -d '<?xml version="1.0" encoding="UTF-8"?><netapp
xmlns="http://www.netapp.com/filer/admin" version="1.21"
vfiler="<vserver-name>"><vserver-get></vserver-get></netapp>'
```

6. Encodieren von Zertifikat, Schlüssel und vertrauenswürdigem CA-Zertifikat mit Base64.

```
base64 -w 0 k8senv.pem >> cert_base64
base64 -w 0 k8senv.key >> key_base64
base64 -w 0 trustedca.pem >> trustedca_base64
```

7. Erstellen Sie das Backend mit den Werten, die aus dem vorherigen Schritt ermittelt wurden.

```

cat cert-backend.json
{
  "version": 1,
  "storageDriverName": "ontap-san",
  "backendName": "SanBackend",
  "managementLIF": "1.2.3.4",
  "svm": "vserver_test",
  "clientCertificate": "Faaaakkkkeeee...Vaaalllluuuuueeee",
  "clientPrivateKey": "LS0tFaKE...0VaLuES0tLS0K",
  "trustedCACertificate": "QNFinf0...SiqOyN",
  "storagePrefix": "myPrefix_"
}

tridentctl create backend -f cert-backend.json -n trident
+-----+-----+-----+-----+
+-----+-----+
|   NAME   | STORAGE DRIVER |           UUID           |
STATE | VOLUMES |
+-----+-----+-----+-----+
+-----+-----+
| SanBackend | ontap-san      | 586b1cd5-8cf8-428d-a76c-2872713612c1 |
online |         0 |
+-----+-----+-----+-----+
+-----+-----+

```

Aktualisieren Sie Authentifizierungsmethoden, oder drehen Sie die Anmeldedaten

Sie können ein vorhandenes Backend aktualisieren, um eine andere Authentifizierungsmethode zu verwenden oder ihre Anmeldedaten zu drehen. Das funktioniert auf beide Arten: Back-Ends, die einen Benutzernamen/ein Passwort verwenden, können aktualisiert werden, um Zertifikate zu verwenden; Back-Ends, die Zertifikate verwenden, können auf Benutzername/Passwort-basiert aktualisiert werden. Dazu müssen Sie die vorhandene Authentifizierungsmethode entfernen und die neue Authentifizierungsmethode hinzufügen. Verwenden Sie dann die aktualisierte Backend.json-Datei, die die erforderlichen Parameter enthält `tridentctl backend update`.

```

cat cert-backend-updated.json
{
"version": 1,
"storageDriverName": "ontap-san",
"backendName": "SanBackend",
"managementLIF": "1.2.3.4",
"svm": "vserver_test",
"username": "vsadmin",
"password": "password",
"storagePrefix": "myPrefix_"
}

#Update backend with tridentctl
tridentctl update backend SanBackend -f cert-backend-updated.json -n
trident
+-----+-----+-----+
+-----+-----+
|   NAME   | STORAGE DRIVER |           UUID           |
STATE | VOLUMES |
+-----+-----+-----+
+-----+-----+
| SanBackend | ontap-san      | 586b1cd5-8cf8-428d-a76c-2872713612c1 |
online |      9 |
+-----+-----+-----+
+-----+-----+

```



Bei der Änderung von Passwörtern muss der Speicheradministrator das Kennwort für den Benutzer auf ONTAP aktualisieren. Auf diese Weise folgt ein Backend-Update. Beim Drehen von Zertifikaten können dem Benutzer mehrere Zertifikate hinzugefügt werden. Das Backend wird dann aktualisiert und verwendet das neue Zertifikat. Danach kann das alte Zertifikat aus dem ONTAP Cluster gelöscht werden.

Durch die Aktualisierung eines Backend wird der Zugriff auf Volumes, die bereits erstellt wurden, nicht unterbrochen, und auch die danach erstellten Volume-Verbindungen werden beeinträchtigt. Ein erfolgreiches Backend-Update zeigt, dass Astra Trident mit dem ONTAP-Backend kommunizieren und zukünftige Volume-Operationen verarbeiten kann.

Verbindungen mit bidirektionalem CHAP authentifizieren

Astra Trident kann iSCSI-Sitzungen mit bidirektionalem CHAP für die authentifizieren `ontap-san` Und `ontap-san-economy` Treiber. Hierfür muss die Aktivierung von erforderlich sein `useCHAP` Option in der Back-End-Definition. Wenn eingestellt auf `true`, Astra Trident konfiguriert die Standard-Initiator-Sicherheit der SVM auf bidirektionales CHAP und setzt den Benutzernamen und die Geheimnisse aus der Backend-Datei. NetApp empfiehlt die Verwendung von bidirektionalem CHAP zur Authentifizierung von Verbindungen. Die folgende Beispielkonfiguration ist verfügbar:

```
---
version: 1
storageDriverName: ontap-san
backendName: ontap_san_chap
managementLIF: 192.168.0.135
svm: ontap_iscsi_svm
useCHAP: true
username: vsadmin
password: password
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSD6cNwxyz
```



Der `useCHAP` Parameter ist eine Boolesche Option, die nur einmal konfiguriert werden kann. Die Standardeinstellung ist „false“. Nachdem Sie die Einstellung auf „true“ gesetzt haben, können Sie sie nicht auf „false“ setzen.

Zusätzlich zu `useCHAP=true`, Das `chapInitiatorSecret`, `chapTargetInitiatorSecret`, `chapTargetUsername`, und `chapUsername` Felder müssen in die Backend-Definition aufgenommen werden. Die Geheimnisse können geändert werden, nachdem ein Backend durch Ausführen erstellt wird `tridentctl update`.

So funktioniert es

Nach Einstellung `useCHAP` Der Storage-Administrator weist Astra Trident an, CHAP im Storage-Back-End zu konfigurieren. Dazu gehört Folgendes:

- Einrichten von CHAP auf der SVM:
 - Wenn der Standard-Initiator-Sicherheitstyp der SVM `none` ist (standardmäßig festgelegt) **und** keine bereits vorhandenen LUNs im Volume vorhanden sind, setzt Astra Trident den Standard-Sicherheitstyp auf `CHAP` Und fahren Sie mit der Konfiguration des CHAP-Initiators und des Zielbenutzernamens und der Schlüssel fort.
 - Wenn die SVM LUNs enthält, aktiviert Astra Trident nicht CHAP auf der SVM. Dadurch wird sichergestellt, dass der Zugriff auf die LUNs, die bereits auf der SVM vorhanden sind, nicht eingeschränkt wird.
- Konfigurieren des CHAP-Initiators und des Ziel-Usernamens und der Schlüssel; diese Optionen müssen in der Back-End-Konfiguration angegeben werden (siehe oben).

Nach der Erstellung des Backend erstellt Astra Trident eine entsprechende `tridentbackend` CRD: Speichert die CHAP-Geheimnisse und Benutzernamen als Kubernetes-Geheimnisse. Alle PVS, die von Astra Trident auf diesem Backend erstellt werden, werden über CHAP gemountet und angeschlossen.

Anmeldedaten rotieren und Back-Ends aktualisieren

Sie können die CHAP-Anmeldeinformationen aktualisieren, indem Sie die CHAP-Parameter im aktualisieren `backend.json` Datei: Dazu müssen die CHAP-Schlüssel aktualisiert und der verwendet werden `tridentctl update` Befehl zum Übergeben dieser Änderungen.



Wenn Sie die CHAP-Schlüssel für ein Backend aktualisieren, müssen Sie verwenden `tridentctl` Um das Backend zu aktualisieren. Aktualisieren Sie die Anmeldeinformationen im Storage-Cluster nicht über die Benutzeroberfläche von CLI/ONTAP, da Astra Trident diese Änderungen nicht übernehmen kann.

```
cat backend-san.json
{
  "version": 1,
  "storageDriverName": "ontap-san",
  "backendName": "ontap_san_chap",
  "managementLIF": "192.168.0.135",
  "svm": "ontap_iscsi_svm",
  "useCHAP": true,
  "username": "vsadmin",
  "password": "password",
  "chapInitiatorSecret": "cl9qxUpDaTeD",
  "chapTargetInitiatorSecret": "rqxigXgkeUpDaTeD",
  "chapTargetUsername": "iJF4heBRT0TCwxyz",
  "chapUsername": "uh2aNCLsd6cNwxyz",
}

./tridentctl update backend ontap_san_chap -f backend-san.json -n trident
+-----+-----+-----+-----+
+-----+-----+
|  NAME          | STORAGE DRIVER |          UUID          |
STATE | VOLUMES |
+-----+-----+-----+-----+
+-----+-----+
| ontap_san_chap | ontap-san      | aa458f3b-ad2d-4378-8a33-1a472ffbeb5c |
online |        7 |
+-----+-----+-----+-----+
+-----+-----+
```

Bestehende Verbindungen bleiben unbeeinträchtigt, sie bleiben auch weiterhin aktiv, wenn die Anmeldedaten vom Astra Trident auf der SVM aktualisiert werden. Neue Verbindungen verwenden die aktualisierten Anmeldedaten und vorhandene Verbindungen bleiben weiterhin aktiv. Wenn Sie alte PVS trennen und neu verbinden, werden sie die aktualisierten Anmeldedaten verwenden.

ONTAP SAN-Konfigurationsoptionen und -Beispiele

Erfahren Sie, wie Sie ONTAP SAN Treiber für Ihre Astra Trident Installation erstellen und verwenden. Dieser Abschnitt enthält Beispiele und Details zur Back-End-Konfiguration für die Zuordnung von Back-Ends zu StorageClasses.

Back-End-Konfigurationsoptionen

Die Back-End-Konfigurationsoptionen finden Sie in der folgenden Tabelle:

| Parameter | Beschreibung | Standard |
|-------------------|--|---|
| version | | Immer 1 |
| storageDriverName | Name des Speichertreibers | ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, ontap-san-economy |
| backendName | Benutzerdefinierter Name oder das Storage-Backend | Treibername + „_“ + DatenLIF |
| managementLIF | <p>Die IP-Adresse einer Cluster- oder SVM-Management-LIF.</p> <p>Es kann ein vollständig qualifizierter Domänenname (FQDN) angegeben werden.</p> <p>Kann so eingestellt werden, dass IPv6-Adressen verwendet werden, wenn Astra Trident mit dem IPv6-Flag installiert wurde. IPv6-Adressen müssen in eckigen Klammern definiert werden, z. B. [28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555].</p> <p>Informationen zur nahtlosen MetroCluster-Umschaltung finden Sie im Beispiel: MetroCluster.</p> | „10.0.0.1“, „[2001:1234:abcd::fefe]“ |
| dataLIF | <p>IP-Adresse des LIF-Protokolls.</p> <p>Nicht für iSCSI angeben. Astra Trident verwendet "ONTAP selektive LUN-Zuordnung" Um die iSCSI LIFs zu ermitteln, die für die Einrichtung einer Multi-Path-Sitzung erforderlich sind. Wenn eine Warnung erzeugt wird dataLIF Ist explizit definiert.</p> <p>Für MetroCluster weglassen. Siehe Beispiel: MetroCluster.</p> | Abgeleitet von SVM |
| svm | <p>Zu verwendende Storage Virtual Machine</p> <p>Für MetroCluster weglassen. Siehe Beispiel: MetroCluster.</p> | Abgeleitet wenn eine SVM managementLIF Angegeben ist |
| useCHAP | <p>Verwenden Sie CHAP, um iSCSI für ONTAP-SAN-Treiber zu authentifizieren [Boolesch].</p> <p>Auf einstellen true Damit Astra Trident bidirektionales CHAP als Standardauthentifizierung für die im Backend angegebene SVM konfiguriert und verwendet. Siehe "Vorbereiten der Konfiguration des Back-End mit ONTAP-SAN-Treibern" Entsprechende Details.</p> | false |

| Parameter | Beschreibung | Standard |
|---------------------------|---|--|
| chapInitiatorSecret | CHAP-Initiatorschlüssel. Erforderlich, wenn useCHAP=true | “ ” |
| labels | Satz willkürlicher JSON-formatierter Etiketten für Volumes | “ ” |
| chapTargetInitiatorSecret | Schlüssel für CHAP-Zielinitiator. Erforderlich, wenn useCHAP=true | “ ” |
| chapUsername | Eingehender Benutzername. Erforderlich, wenn useCHAP=true | “ ” |
| chapTargetUsername | Zielbenutzername. Erforderlich, wenn useCHAP=true | “ ” |
| clientCertificate | Base64-codierter Wert des Clientzertifikats. Wird für zertifikatbasierte Authentifizierung verwendet | “ ” |
| clientPrivateKey | Base64-kodierte Wert des privaten Client-Schlüssels. Wird für zertifikatbasierte Authentifizierung verwendet | “ ” |
| trustedCACertificate | Base64-kodierte Wert des vertrauenswürdigen CA-Zertifikats. Optional Wird für die zertifikatbasierte Authentifizierung verwendet. | “ ” |
| username | Benutzername für die Kommunikation mit dem ONTAP Cluster erforderlich. Wird für die Anmeldeinformationsbasierte Authentifizierung verwendet. | “ ” |
| password | Passwort, das für die Kommunikation mit dem ONTAP Cluster erforderlich ist. Wird für die Anmeldeinformationsbasierte Authentifizierung verwendet. | “ ” |
| svm | Zu verwendende Storage Virtual Machine | Abgeleitet wenn eine SVM managementLIF Angegeben ist |
| storagePrefix | Das Präfix wird beim Bereitstellen neuer Volumes in der SVM verwendet. Kann später nicht mehr geändert werden. Um diesen Parameter zu aktualisieren, müssen Sie ein neues Backend erstellen. | trident |
| limitAggregateUsage | Bereitstellung fehlgeschlagen, wenn die Nutzung über diesem Prozentsatz liegt. Wenn Sie ein Amazon FSX für das NetApp ONTAP-Back-End verwenden, geben Sie diese bitte nicht an limitAggregateUsage. Die vorhanden fsxadmin Und vsadmin Enthalten Sie nicht die erforderlichen Berechtigungen, um die Aggregatnutzung abzurufen und sie mit Astra Trident zu begrenzen. | “ ” (nicht standardmäßig durchgesetzt) |

| Parameter | Beschreibung | Standard |
|-----------------|---|-----------------------------------|
| limitVolumeSize | Bereitstellung fehlgeschlagen, wenn die angeforderte Volume-Größe über diesem Wert liegt. Schränkt auch die maximale Größe der Volumes ein, die es für qtrees und LUNs managt. | „ (standardmäßig nicht erzwungen) |
| lunsPerFlexvol | Die maximale Anzahl an LUNs pro FlexVol muss im Bereich [50, 200] liegen. | 100 |
| debugTraceFlags | Fehler-Flags bei der Fehlerbehebung beheben. Beispiel, {„API“:false, „method“:true} Verwenden Sie diese Funktion nur, wenn Sie eine Fehlerbehebung durchführen und einen detaillierten Protokollauszug benötigen. | null |
| useREST | Boolescher Parameter zur Verwendung von ONTAP REST-APIs. Technische Vorschau useREST Wird als Tech-Vorschau bereitgestellt , das für Testumgebungen und nicht für Produktions-Workloads empfohlen wird. Wenn eingestellt auf <code>true</code> , Astra Trident wird ONTAP REST APIs zur Kommunikation mit dem Backend verwenden. Diese Funktion erfordert ONTAP 9.11.1 und höher. Darüber hinaus muss die verwendete ONTAP-Login-Rolle Zugriff auf den haben <code>ontap</code> Applikation. Dies wird durch die vordefinierte zufrieden <code>vsadmin</code> Und <code>cluster-admin</code> Rollen: useREST Wird mit MetroCluster nicht unterstützt. | false |

Back-End-Konfigurationsoptionen für die Bereitstellung von Volumes

Sie können die Standardbereitstellung mit diesen Optionen im `steuern defaults` Abschnitt der Konfiguration. Ein Beispiel finden Sie unten in den Konfigurationsbeispielen.

| Parameter | Beschreibung | Standard |
|-----------------|--|----------|
| spaceAllocation | Speicherplatzzuweisung für LUNs | „Wahr“ |
| spaceReserve | Modus für Speicherplatzreservierung; „none“ (Thin) oder „Volume“ (Thick) | „Keine“ |
| snapshotPolicy | Die Snapshot-Richtlinie zu verwenden | „Keine“ |

| Parameter | Beschreibung | Standard |
|-------------------|---|---|
| qosPolicy | <p>QoS-Richtliniengruppe zur Zuweisung für erstellte Volumes Wählen Sie eine der qosPolicy oder adaptiveQosPolicy pro Storage Pool/Backend.</p> <p>Die Verwendung von QoS Policy Groups mit Astra Trident erfordert ONTAP 9.8 oder höher. Wir empfehlen die Verwendung einer nicht gemeinsam genutzten QoS-Richtliniengruppe und stellen sicher, dass die Richtliniengruppe auf jede Komponente einzeln angewendet wird. Eine Richtliniengruppe für Shared QoS führt zur Durchsetzung der Obergrenze für den Gesamtdurchsatz aller Workloads.</p> | “ ” |
| adaptiveQosPolicy | <p>Adaptive QoS-Richtliniengruppe mit Zuordnung für erstellte Volumes Wählen Sie eine der qosPolicy oder adaptiveQosPolicy pro Storage Pool/Backend</p> | “ ” |
| snapshotReserve | <p>Prozentsatz des für Snapshots reservierten Volumes</p> | „0“ wenn snapshotPolicy Ist „keine“, andernfalls „“ |
| splitOnClone | <p>Teilen Sie einen Klon bei der Erstellung von seinem übergeordneten Objekt auf</p> | „Falsch“ |
| encryption | <p>Aktivieren Sie NetApp Volume Encryption (NVE) auf dem neuen Volume, standardmäßig aktiviert <code>false</code>. NVE muss im Cluster lizenziert und aktiviert sein, damit diese Option verwendet werden kann.</p> <p>Wenn NAE auf dem Backend aktiviert ist, wird jedes im Astra Trident bereitgestellte Volume NAE aktiviert.</p> <p>Weitere Informationen finden Sie unter: "Astra Trident arbeitet mit NVE und NAE zusammen".</p> | „Falsch“ |
| luksEncryption | <p>Aktivieren Sie die LUKS-Verschlüsselung. Siehe "Linux Unified Key Setup (LUKS) verwenden".</p> | “ ” |

| Parameter | Beschreibung | Standard |
|---------------|--|--|
| securityStyle | Sicherheitstyp für neue Volumes | unix |
| tieringPolicy | Tiering-Richtlinie, die zu „keinen“ verwendet wird | „Nur snapshot“ für eine SVM-DR-Konfiguration vor ONTAP 9.5 |

Beispiele für die Volume-Bereitstellung

Hier ein Beispiel mit definierten Standardwerten:

```

---
version: 1
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: trident_svm
username: admin
password: <password>
labels:
  k8scluster: dev2
  backend: dev2-sanbackend
storagePrefix: alternate-trident
debugTraceFlags:
  api: false
  method: true
defaults:
  spaceReserve: volume
  qosPolicy: standard
  spaceAllocation: 'false'
  snapshotPolicy: default
  snapshotReserve: '10'

```



Für alle mit dem erstellten Volumes `ontap-san` Treiber: Astra Trident fügt der FlexVol zusätzliche Kapazität von 10 % hinzu, um die LUN-Metadaten zu bewältigen. Die LUN wird genau mit der Größe bereitgestellt, die der Benutzer in der PVC anfordert. Astra Trident fügt 10 Prozent zum FlexVol hinzu (wird in ONTAP als verfügbare Größe dargestellt). Benutzer erhalten jetzt die Menge an nutzbarer Kapazität, die sie angefordert haben. Diese Änderung verhindert auch, dass LUNs schreibgeschützt werden, sofern der verfügbare Speicherplatz nicht vollständig genutzt wird. Dies gilt nicht für die Wirtschaft von `ontap-san`.

Für Back-Ends, die definieren `snapshotReserve`, Astra Trident berechnet die Größe der Volumes wie folgt:

$$\text{Total volume size} = [(\text{PVC requested size}) / (1 - (\text{snapshotReserve percentage} / 100))] * 1.1$$

Das 1.1 ist der zusätzliche 10-Prozent-Astra Trident fügt dem FlexVol hinzu, um die LUN-Metadaten zu bewältigen. Für `snapshotReserve = 5 %`, und die PVC-Anforderung = 5 gib, die Gesamtgröße des Volumes

beträgt 5,79 gib und die verfügbare Größe 5,5 gib. Der `volume show` Der Befehl sollte Ergebnisse anzeigen, die diesem Beispiel ähnlich sind:

| Vserver | Volume | Aggregate | State | Type | Size | Available | Used% |
|---------|--------|---|--------|------|--------|-----------|-------|
| | | _pvc_89f1c156_3801_4de4_9f9d_034d54c395f4 | online | RW | 10GB | 5.00GB | 0% |
| | | _pvc_e42ec6fe_3baa_4af6_996d_134adbbb8e6d | online | RW | 5.79GB | 5.50GB | 0% |
| | | _pvc_e8372153_9ad9_474a_951a_08ae15e1c0ba | online | RW | 1GB | 511.8MB | 0% |

3 entries were displayed.

Die Größenanpassung ist derzeit die einzige Möglichkeit, die neue Berechnung für ein vorhandenes Volume zu verwenden.

Minimale Konfigurationsbeispiele

Die folgenden Beispiele zeigen grundlegende Konfigurationen, bei denen die meisten Parameter standardmäßig belassen werden. Dies ist der einfachste Weg, ein Backend zu definieren.



Wenn Sie Amazon FSX auf NetApp ONTAP mit Astra Trident verwenden, empfehlen wir, DNS-Namen für LIFs anstelle von IP-Adressen anzugeben.

Beispiel: ONTAP SAN

Dies ist eine grundlegende Konfiguration mit dem `ontap-san` Treiber.

```
---
version: 1
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_iscsi
labels:
  k8scluster: test-cluster-1
  backend: testcluster1-sanbackend
username: vsadmin
password: <password>
```

Beispiel für die SAN-Ökonomie von ONTAP

```
---  
version: 1  
storageDriverName: ontap-san-economy  
managementLIF: 10.0.0.1  
svm: svm_iscsi_eco  
username: vsadmin  
password: <password>
```

Beispiel: MetroCluster

Sie können das Backend so konfigurieren, dass die Backend-Definition nach Umschaltung und einem Wechsel während nicht manuell aktualisiert werden muss "[SVM-Replizierung und Recovery](#)".

Für nahtloses Switchover und Switchback geben Sie die SVM über an `managementLIF` Und lassen Sie die aus `dataLIF` Und `svm` Parameter. Beispiel:

```
---  
version: 1  
storageDriverName: ontap-san  
managementLIF: 192.168.1.66  
username: vsadmin  
password: password
```

Beispiel für die zertifikatbasierte Authentifizierung

In diesem Beispiel der Grundkonfiguration `clientCertificate`, `clientPrivateKey`, und `trustedCACertificate` (Optional, wenn Sie eine vertrauenswürdige CA verwenden) werden ausgefüllt `backend.json` Und nehmen Sie die base64-kodierten Werte des Clientzertifikats, des privaten Schlüssels und des vertrauenswürdigen CA-Zertifikats.

```
---
version: 1
storageDriverName: ontap-san
backendName: DefaultSANBackend
managementLIF: 10.0.0.1
svm: svm_iscsi
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSD6cNwxyz
clientCertificate: ZXR0ZXJwYXB...ICMgJ3BhcGVyc2
clientPrivateKey: vciwKIyAgZG...0cnksIGRlc2NyaX
trustedCACertificate: zcyBbaG...b3Igb3duIGNsYXNz
```

Beispiele für bidirektionales CHAP

Diese Beispiele erstellen ein Backend mit `useCHAP` Auf einstellen `true`.

Beispiel für ONTAP-SAN-CHAP

```
---
version: 1
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_iscsi
labels:
  k8scluster: test-cluster-1
  backend: testcluster1-sanbackend
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSD6cNwxyz
username: vsadmin
password: <password>
```

Beispiel für ONTAP SAN Economy CHAP

```
---
version: 1
storageDriverName: ontap-san-economy
managementLIF: 10.0.0.1
svm: svm_iscsi_eco
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSD6cNwxyz
username: vsadmin
password: <password>
```

Beispiele für Back-Ends mit virtuellen Pools

In diesen Beispiel-Back-End-Definitionsdateien werden spezifische Standardwerte für alle Speicherpools festgelegt, z. B. `spaceReserve` Bei keiner, `spaceAllocation` Bei false, und `encryption` Bei false. Die virtuellen Pools werden im Abschnitt Speicher definiert.

Astra Trident bestimmt die Bereitstellungsetiketten im Feld „Kommentare“. Kommentare werden auf dem FlexVol gesetzt. Astra Trident kopiert alle Labels auf einem virtuellen Pool auf das Storage-Volume während der Bereitstellung. Storage-Administratoren können Labels je virtuellen Pool definieren und Volumes nach Label gruppieren.

In diesen Beispielen legen einige Speicherpools eigene fest `spaceReserve`, `spaceAllocation`, und `encryption` Werte und einige Pools überschreiben die Standardwerte.

Beispiel: ONTAP SAN



```
---
version: 1
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_iscsi
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSD6cNwxyz
username: vsadmin
password: <password>
defaults:
  spaceAllocation: 'false'
  encryption: 'false'
  qosPolicy: standard
labels:
  store: san_store
  kubernetes-cluster: prod-cluster-1
region: us_east_1
storage:
- labels:
  protection: gold
  creditpoints: '40000'
  zone: us_east_1a
  defaults:
    spaceAllocation: 'true'
    encryption: 'true'
    adaptiveQosPolicy: adaptive-extreme
- labels:
  protection: silver
  creditpoints: '20000'
  zone: us_east_1b
  defaults:
    spaceAllocation: 'false'
    encryption: 'true'
    qosPolicy: premium
- labels:
  protection: bronze
  creditpoints: '5000'
  zone: us_east_1c
  defaults:
    spaceAllocation: 'true'
    encryption: 'false'
```

Beispiel für die SAN-Ökonomie von ONTAP

```
---
version: 1
storageDriverName: ontap-san-economy
managementLIF: 10.0.0.1
svm: svm_iscsi_eco
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSD6cNwxyz
username: vsadmin
password: <password>
defaults:
  spaceAllocation: 'false'
  encryption: 'false'
labels:
  store: san_economy_store
region: us_east_1
storage:
- labels:
  app: oracledb
  cost: '30'
  zone: us_east_1a
  defaults:
    spaceAllocation: 'true'
    encryption: 'true'
- labels:
  app: postgresdb
  cost: '20'
  zone: us_east_1b
  defaults:
    spaceAllocation: 'false'
    encryption: 'true'
- labels:
  app: mysqldb
  cost: '10'
  zone: us_east_1c
  defaults:
    spaceAllocation: 'true'
    encryption: 'false'
- labels:
  department: legal
  creditpoints: '5000'
  zone: us_east_1c
```

```
defaults:
  spaceAllocation: 'true'
  encryption: 'false'
```

Back-Ends StorageClasses zuordnen

Die folgenden StorageClass-Definitionen finden Sie im [Beispiele für Back-Ends mit virtuellen Pools](#).

Verwenden der `parameters.selector` Jede StorageClass ruft auf, welche virtuellen Pools zum Hosten eines Volumes verwendet werden können. Auf dem Volume werden die Aspekte im ausgewählten virtuellen Pool definiert.

- Der `protection-gold` StorageClass wird dem ersten virtuellen Pool in zugeordnet `ontap-san` Back-End: Dies ist der einzige Pool mit Gold-Level-Schutz.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-gold
provisioner: netapp.io/trident
parameters:
  selector: "protection=gold"
  fsType: "ext4"
```

- Der `protection-not-gold` StorageClass wird dem zweiten und dritten virtuellen Pool in zugeordnet `ontap-san` Back-End: Dies sind die einzigen Pools, die ein anderes Schutzniveau als Gold bieten.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-not-gold
provisioner: netapp.io/trident
parameters:
  selector: "protection!=gold"
  fsType: "ext4"
```

- Der `app-mysqldb` StorageClass wird dem dritten virtuellen Pool in zugeordnet `ontap-san-economy` Back-End: Dies ist der einzige Pool, der Storage-Pool-Konfiguration für die `mysqldb`-App bietet.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: app-mysqldb
provisioner: netapp.io/trident
parameters:
  selector: "app=mysqldb"
  fsType: "ext4"
```

- Der `protection-silver-creditpoints-20k` StorageClass wird dem zweiten virtuellen Pool in zugeordnet `ontap-san` Back-End: Dies ist der einzige Pool mit Silber-Level-Schutz und 20000 Kreditpunkte.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-silver-creditpoints-20k
provisioner: netapp.io/trident
parameters:
  selector: "protection=silver; creditpoints=20000"
  fsType: "ext4"
```

- Der `creditpoints-5k` StorageClass wird dem dritten virtuellen Pool in zugeordnet `ontap-san` Back-End und der vierte virtuelle Pool im `ontap-san-economy` Back-End: Dies sind die einzigen Poolangebote mit 5000 Kreditpunkten.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: creditpoints-5k
provisioner: netapp.io/trident
parameters:
  selector: "creditpoints=5000"
  fsType: "ext4"
```

Astra Trident entscheidet, welcher virtuelle Pool ausgewählt wird und stellt sicher, dass die Storage-Anforderungen erfüllt werden.

ONTAP-NAS-Treiber

Übersicht über den ONTAP NAS-Treiber

Erfahren Sie mehr über die Konfiguration eines ONTAP-Backend mit ONTAP- und Cloud Volumes ONTAP-NAS-Treibern.

Details zum ONTAP NAS-Treiber

Astra Trident bietet die folgenden NAS-Storage-Treiber für die Kommunikation mit dem ONTAP Cluster. Unterstützte Zugriffsmodi sind: *ReadWriteOnce* (RWO), *ReadOnly Many* (ROX), *ReadWriteMany* (RWX), *ReadWriteOncePod* (RWOP).



Wenn Sie Astra Control für Schutz, Recovery und Mobilität verwenden, lesen Sie bitte [Treiberkompatibilität bei Astra Control](#).

| Treiber | Protokoll | VolumeModus | Unterstützte Zugriffsmodi | Unterstützte Filesysteme |
|---------------------|------------|-------------|---------------------------|--------------------------|
| ontap-nas | NFS SMB | Dateisystem | RWO, ROX, RWX, RWOP | „, nfs, smb |
| ontap-nas-economy | NFS SMB | Dateisystem | RWO, ROX, RWX, RWOP | „, nfs, smb |
| ontap-nas-flexgroup | NFS SMB | Dateisystem | RWO, ROX, RWX, RWOP | „, nfs, smb |

Treiberkompatibilität bei Astra Control

Astra Control bietet nahtlosen Schutz, Disaster Recovery und Mobilität (Verschieben von Volumes zwischen Kubernetes Clustern) für Volumes, die mit der erstellt wurden `ontap-nas`, `ontap-nas-flexgroup`, und `ontap-san` Treiber. Siehe "[Voraussetzungen für die Astra Control Replikation](#)" Entsprechende Details.



- Nutzung `ontap-san-economy` Nur wenn die Nutzungszahl für persistente Volumes voraussichtlich höher ist als "[Unterstützte ONTAP-Volume-Größen](#)".
- Nutzung `ontap-nas-economy` Nur wenn die Nutzungszahl für persistente Volumes voraussichtlich höher ist als "[Unterstützte ONTAP-Volume-Größen](#)" Und das `ontap-san-economy` Treiber kann nicht verwendet werden.
- Verwenden Sie ihn nicht `ontap-nas-economy` Wenn Sie die Notwendigkeit von Datensicherung, Disaster Recovery oder Mobilität erwarten.

Benutzerberechtigungen

Astra Trident erwartet, dass er entweder als ONTAP- oder SVM-Administrator ausgeführt wird, in der Regel mit dem `admin` Cluster-Benutzer oder ein `vsadmin` SVM-Benutzer oder ein Benutzer mit einem anderen Namen und derselben Rolle.

Astra Trident erwartet, dass bei Amazon FSX für Implementierungen von NetApp ONTAP, über das Cluster entweder als ONTAP- oder SVM-Administrator ausgeführt wird `fsxadmin` Benutzer oder A `vsadmin` SVM-Benutzer oder ein Benutzer mit einem anderen Namen und derselben Rolle. Der `fsxadmin` Der Benutzer ist ein eingeschränkter Ersatz für den Cluster-Admin-Benutzer.



Wenn Sie den verwenden `limitAggregateUsage` Parameter, Berechtigungen für Cluster-Admin sind erforderlich. Bei der Verwendung von Amazon FSX für NetApp ONTAP mit Astra Trident, das `limitAggregateUsage` Der Parameter funktioniert nicht mit dem `vsadmin` Und `fsxadmin` Benutzerkonten. Der Konfigurationsvorgang schlägt fehl, wenn Sie diesen Parameter angeben.

Es ist zwar möglich, eine restriktivere Rolle in ONTAP zu erstellen, die ein Trident-Treiber verwenden kann, wir empfehlen sie jedoch nicht. Bei den meisten neuen Versionen von Trident sind zusätzliche APIs erforderlich, die berücksichtigt werden müssten, was Upgrades schwierig und fehleranfällig macht.

Bereiten Sie sich auf die Konfiguration eines Backend mit ONTAP-NAS-Treibern vor

Verstehen Sie die Anforderungen, Authentifizierungsoptionen und Exportrichtlinien für die Konfiguration eines ONTAP-Backends mit ONTAP-NAS-Treibern.

Anforderungen

- Für alle ONTAP Back-Ends benötigt Astra Trident mindestens ein Aggregat, das der SVM zugewiesen ist.
- Sie können mehrere Treiber ausführen und Speicherklassen erstellen, die auf den einen oder den anderen zeigen. Beispielsweise könnten Sie eine Gold-Klasse konfigurieren, die den verwendet `ontap-nas` Fahrer und eine Bronze-Klasse, die den verwendet `ontap-nas-economy` Eins.
- Alle Kubernetes-Worker-Nodes müssen über die entsprechenden NFS-Tools verfügen. Siehe "[Hier](#)" Entnehmen.
- Astra Trident unterstützt SMB Volumes, die nur auf Windows Nodes laufenden Pods gemountet werden. Siehe [Vorbereitung zur Bereitstellung von SMB Volumes](#) Entsprechende Details.

Authentifizieren Sie das ONTAP-Backend

Astra Trident bietet zwei Arten der Authentifizierung eines ONTAP-Backend.

- Anmeldeinformationsbasiert: Benutzername und Passwort für einen ONTAP-Benutzer mit den erforderlichen Berechtigungen. Es wird empfohlen, eine vordefinierte Sicherheits-Login-Rolle zu verwenden, wie z. B. `admin` Oder `vsadmin` Für maximale Kompatibilität mit ONTAP Versionen.
- Zertifikatsbasiert: Astra Trident kann auch mit einem ONTAP Cluster kommunizieren. Verwenden Sie dazu ein Zertifikat, das auf dem Backend installiert ist. Hier muss die Backend-Definition Base64-kodierte Werte des Client-Zertifikats, des Schlüssels und des vertrauenswürdigen CA-Zertifikats enthalten, sofern verwendet (empfohlen).

Sie können vorhandene Back-Ends aktualisieren, um zwischen auf Anmeldeinformationen basierenden und zertifikatbasierten Methoden zu verschieben. Es wird jedoch immer nur eine Authentifizierungsmethode unterstützt. Um zu einer anderen Authentifizierungsmethode zu wechseln, müssen Sie die vorhandene Methode von der Backend-Konfiguration entfernen.



Wenn Sie versuchen, **sowohl Anmeldeinformationen als auch Zertifikate** bereitzustellen, schlägt die Backend-Erstellung mit einem Fehler fehl, dass mehr als eine Authentifizierungsmethode in der Konfigurationsdatei angegeben wurde.

Aktivieren Sie die Anmeldeinformationsbasierte Authentifizierung

Astra Trident erfordert die Zugangsdaten für einen Administrator mit SVM-Umfang/Cluster-Umfang, um mit dem Backend von ONTAP zu kommunizieren. Es wird empfohlen, die Standard-vordefinierten Rollen wie zu

verwenden `admin` Oder `vsadmin`. So ist gewährleistet, dass die Kompatibilität mit künftigen ONTAP Versionen gewährleistet ist, die FunktionsAPIs der künftigen Astra Trident Versionen bereitstellen können. Eine benutzerdefinierte Sicherheits-Login-Rolle kann mit Astra Trident erstellt und verwendet werden, wird aber nicht empfohlen.

Eine Beispiel-Back-End-Definition sieht folgendermaßen aus:

YAML

```
---
version: 1
backendName: ExampleBackend
storageDriverName: ontap-nas
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
svm: svm_nfs
username: vsadmin
password: password
```

JSON

```
{
  "version": 1,
  "backendName": "ExampleBackend",
  "storageDriverName": "ontap-nas",
  "managementLIF": "10.0.0.1",
  "dataLIF": "10.0.0.2",
  "svm": "svm_nfs",
  "username": "vsadmin",
  "password": "password"
}
```

Beachten Sie, dass die Backend-Definition der einzige Ort ist, an dem die Anmeldeinformationen im reinen Text gespeichert werden. Nach der Erstellung des Backend werden Benutzernamen/Passwörter mit Base64 codiert und als Kubernetes Secrets gespeichert. Die Erstellung/Aktualisierung eines Backend ist der einzige Schritt, der Kenntnisse der Anmeldeinformationen erfordert. Daher ist dieser Vorgang nur für Administratoren und wird vom Kubernetes-/Storage-Administrator ausgeführt.

Aktivieren Sie die zertifikatbasierte Authentifizierung

Neue und vorhandene Back-Ends können ein Zertifikat verwenden und mit dem ONTAP-Back-End kommunizieren. In der Backend-Definition sind drei Parameter erforderlich.

- `ClientCertificate`: Base64-codierter Wert des Clientzertifikats.
- `ClientPrivateKey`: Base64-kodierte Wert des zugeordneten privaten Schlüssels.
- `Trusted CACertificate`: Base64-codierter Wert des vertrauenswürdigen CA-Zertifikats. Bei Verwendung einer vertrauenswürdigen CA muss dieser Parameter angegeben werden. Dies kann ignoriert werden,

wenn keine vertrauenswürdige CA verwendet wird.

Ein typischer Workflow umfasst die folgenden Schritte.

Schritte

1. Erzeugen eines Clientzertifikats und eines Schlüssels. Legen Sie beim Generieren den allgemeinen Namen (CN) für den ONTAP-Benutzer fest, der sich authentifizieren soll als.

```
openssl req -x509 -nodes -days 1095 -newkey rsa:2048 -keyout k8senv.key  
-out k8senv.pem -subj "/C=US/ST=NC/L=RTP/O=NetApp/CN=vsadmin"
```

2. Fügen Sie dem ONTAP-Cluster ein vertrauenswürdigen CA-Zertifikat hinzu. Dies kann möglicherweise bereits vom Storage-Administrator übernommen werden. Ignorieren, wenn keine vertrauenswürdige CA verwendet wird.

```
security certificate install -type server -cert-name <trusted-ca-cert-  
name> -vserver <vserver-name>  
ssl modify -vserver <vserver-name> -server-enabled true -client-enabled  
true -common-name <common-name> -serial <SN-from-trusted-CA-cert> -ca  
<cert-authority>
```

3. Installieren Sie das Client-Zertifikat und den Schlüssel (von Schritt 1) auf dem ONTAP-Cluster.

```
security certificate install -type client-ca -cert-name <certificate-  
name> -vserver <vserver-name>  
security ssl modify -vserver <vserver-name> -client-enabled true
```

4. Bestätigen Sie, dass die ONTAP-Sicherheitsanmeldungsrolle unterstützt wird `cert` Authentifizierungsmethode.

```
security login create -user-or-group-name vsadmin -application ontapi  
-authentication-method cert -vserver <vserver-name>  
security login create -user-or-group-name vsadmin -application http  
-authentication-method cert -vserver <vserver-name>
```

5. Testen Sie die Authentifizierung mithilfe des generierten Zertifikats. `<ONTAP Management LIF>` und `<vServer Name>` durch Management-LIF-IP und SVM-Namen ersetzen. Sie müssen sicherstellen, dass die Service-Richtlinie für das LIF auf festgelegt ist `default-data-management`.

```
curl -X POST -Lk https://<ONTAP-Management-
LIF>/servlets/netapp.servlets.admin.XMLrequest_filer --key k8senv.key
--cert ~/k8senv.pem -d '<?xml version="1.0" encoding="UTF-8"?><netapp
xmlns="http://www.netapp.com/filer/admin" version="1.21"
vfiler="<vserver-name>"><vserver-get></vserver-get></netapp>'
```

6. Encodieren von Zertifikat, Schlüssel und vertrauenswürdigen CA-Zertifikat mit Base64.

```
base64 -w 0 k8senv.pem >> cert_base64
base64 -w 0 k8senv.key >> key_base64
base64 -w 0 trustedca.pem >> trustedca_base64
```

7. Erstellen Sie das Backend mit den Werten, die aus dem vorherigen Schritt ermittelt wurden.

```
cat cert-backend-updated.json
{
  "version": 1,
  "storageDriverName": "ontap-nas",
  "backendName": "NasBackend",
  "managementLIF": "1.2.3.4",
  "dataLIF": "1.2.3.8",
  "svm": "vserver_test",
  "clientCertificate": "Faaaakkkkeeee...Vaaalllluuueeeee",
  "clientPrivateKey": "LS0tFaKE...0VaLuES0tLS0K",
  "storagePrefix": "myPrefix_"
}

#Update backend with tridentctl
tridentctl update backend NasBackend -f cert-backend-updated.json -n
trident

+-----+-----+-----+-----+
+-----+-----+
|   NAME   | STORAGE DRIVER |           UUID           |
STATE | VOLUMES |
+-----+-----+-----+-----+
+-----+-----+
| NasBackend | ontap-nas      | 98e19b74-aec7-4a3d-8dcf-128e5033b214 |
online |          9 |
+-----+-----+-----+-----+
+-----+-----+

```

Aktualisieren Sie Authentifizierungsmethoden, oder drehen Sie die Anmeldedaten

Sie können ein vorhandenes Backend aktualisieren, um eine andere Authentifizierungsmethode zu verwenden oder ihre Anmeldedaten zu drehen. Das funktioniert auf beide Arten: Back-Ends, die einen Benutzernamen/ein Passwort verwenden, können aktualisiert werden, um Zertifikate zu verwenden; Back-Ends, die Zertifikate verwenden, können auf Benutzername/Passwort-basiert aktualisiert werden. Dazu müssen Sie die vorhandene Authentifizierungsmethode entfernen und die neue Authentifizierungsmethode hinzufügen. Verwenden Sie dann die aktualisierte Backend.json-Datei, die die erforderlichen Parameter enthält `tridentctl update backend`.

```
cat cert-backend-updated.json
{
  "version": 1,
  "storageDriverName": "ontap-nas",
  "backendName": "NasBackend",
  "managementLIF": "1.2.3.4",
  "dataLIF": "1.2.3.8",
  "svm": "vserver_test",
  "username": "vsadmin",
  "password": "password",
  "storagePrefix": "myPrefix_"
}

#Update backend with tridentctl
tridentctl update backend NasBackend -f cert-backend-updated.json -n
trident
+-----+-----+-----+-----+
+-----+-----+
|   NAME   | STORAGE DRIVER |           UUID           |
STATE | VOLUMES |
+-----+-----+-----+-----+
+-----+-----+
| NasBackend | ontap-nas      | 98e19b74-aec7-4a3d-8dcf-128e5033b214 |
online |          9 |
+-----+-----+-----+-----+
+-----+-----+

```



Bei der Änderung von Passwörtern muss der Speicheradministrator das Kennwort für den Benutzer auf ONTAP aktualisieren. Auf diese Weise folgt ein Backend-Update. Beim Drehen von Zertifikaten können dem Benutzer mehrere Zertifikate hinzugefügt werden. Das Backend wird dann aktualisiert und verwendet das neue Zertifikat. Danach kann das alte Zertifikat aus dem ONTAP Cluster gelöscht werden.

Durch die Aktualisierung eines Backend wird der Zugriff auf Volumes, die bereits erstellt wurden, nicht unterbrochen, und auch die danach erstellten Volume-Verbindungen werden beeinträchtigt. Ein erfolgreiches Backend-Update zeigt, dass Astra Trident mit dem ONTAP-Backend kommunizieren und zukünftige Volume-Operationen verarbeiten kann.

Management der NFS-Exportrichtlinien

Astra Trident verwendet NFS-Exportrichtlinien, um den Zugriff auf die Volumes zu kontrollieren, die er bereitstellt.

Astra Trident bietet zwei Optionen für die Arbeit mit Exportrichtlinien:

- Astra Trident kann die Exportrichtlinie selbst dynamisch managen. In diesem Betriebsmodus spezifiziert der Storage-Administrator eine Liste mit CIDR-Blöcken, die zulässige IP-Adressen darstellen. Astra Trident fügt automatisch Node-IPs hinzu, die in diese Bereiche fallen, zur Exportrichtlinie hinzu. Wenn keine CIDRs angegeben werden, wird alternativ jede auf den Knoten gefundene globale Unicast-IP mit globalem Umfang zur Exportrichtlinie hinzugefügt.
- Storage-Administratoren können eine Exportrichtlinie erstellen und Regeln manuell hinzufügen. Astra Trident verwendet die Standard-Exportrichtlinie, es sei denn, in der Konfiguration ist ein anderer Name der Exportrichtlinie angegeben.

Dynamisches Managen von Exportrichtlinien

Astra Trident bietet die Möglichkeit, Richtlinien für den Export von ONTAP Back-Ends dynamisch zu managen. So kann der Storage-Administrator einen zulässigen Adressraum für Worker-Node-IPs festlegen, anstatt explizite Regeln manuell zu definieren. Dies vereinfacht das Management von Exportrichtlinien erheblich. Änderungen der Exportrichtlinie erfordern keine manuellen Eingriffe des Storage-Clusters mehr. Darüber hinaus hilft dies, den Zugriff auf den Storage-Cluster nur auf Worker-Nodes mit IPs im angegebenen Bereich zu beschränken, was ein fein abgestimmtes und automatisiertes Management unterstützt.



Verwenden Sie keine Network Address Translation (NAT), wenn Sie dynamische Exportrichtlinien verwenden. Bei NAT erkennt der Speicher-Controller die Frontend-NAT-Adresse und nicht die tatsächliche IP-Host-Adresse, so dass der Zugriff verweigert wird, wenn in den Exportregeln keine Übereinstimmung gefunden wird.

Beispiel

Es müssen zwei Konfigurationsoptionen verwendet werden. Hier ist eine Beispiel-Backend-Definition:

```
---
version: 1
storageDriverName: ontap-nas
backendName: ontap_nas_auto_export
managementLIF: 192.168.0.135
svm: svm1
username: vsadmin
password: password
autoExportCIDRs:
- 192.168.0.0/24
autoExportPolicy: true
```



Wenn Sie diese Funktion verwenden, müssen Sie sicherstellen, dass für die Root-Verbindung in Ihrer SVM eine zuvor erstellte Exportrichtlinie mit einer Exportregel vorhanden ist, die den CIDR-Block des Nodes zulässt (z. B. die standardmäßige Exportrichtlinie). Folgen Sie stets den von NetApp empfohlenen Best Practices, um eine SVM für Astra Trident zu zuweisen.

Hier ist eine Erklärung, wie diese Funktion funktioniert, anhand des obigen Beispiels:

- `autoExportPolicy` Ist auf festgelegt `true`. Dies zeigt an, dass Astra Trident eine Exportrichtlinie für den erstellen wird `svm1` SVM und das Hinzufügen und Löschen von Regeln mit behandeln `autoExportCIDRs` Adressblöcke. Beispiel: Ein Backend mit UUID `403b5326-8482-40db-96d0-d83fb3f4daec` und `autoExportPolicy` Auf einstellen `true` Erstellt eine Exportrichtlinie mit dem Namen `trident-403b5326-8482-40db-96d0-d83fb3f4daec` Auf der SVM.
- `autoExportCIDRs` Enthält eine Liste von Adressblöcken. Dieses Feld ist optional und standardmäßig `[„0.0.0.0/0“, „:/0“]`. Falls nicht definiert, fügt Astra Trident alle Unicast-Adressen mit globalem Umfang hinzu, die auf den Worker-Nodes gefunden wurden.

In diesem Beispiel ist der `192.168.0.0/24` Adressbereich wird bereitgestellt. Das zeigt an, dass die Kubernetes-Node-IPs, die in diesen Adressbereich fallen, der vom Astra Trident erstellten Exportrichtlinie hinzugefügt werden. Wenn Astra Trident einen Knoten registriert, auf dem er ausgeführt wird, ruft er die IP-Adressen des Knotens ab und überprüft sie auf die in angegebenen Adressblöcke `autoExportCIDRs`. Nach dem Filtern der IPs erstellt Astra Trident Regeln für die Exportrichtlinie für die erkannte Client-IPs. Dabei gilt für jeden Node eine Regel, die er identifiziert.

Sie können aktualisieren `autoExportPolicy` Und `autoExportCIDRs` Für Back-Ends, nachdem Sie sie erstellt haben. Sie können neue CIDRs für ein Backend anhängen, das automatisch verwaltet wird oder vorhandene CIDRs löschen. Beim Löschen von CIDRs Vorsicht walten lassen, um sicherzustellen, dass vorhandene Verbindungen nicht unterbrochen werden. Sie können auch wählen, zu deaktivieren `autoExportPolicy` Für ein Backend und kehren Sie zu einer manuell erstellten Exportrichtlinie zurück. Dazu muss die Einstellung festgelegt werden `exportPolicy` Parameter in Ihrer Backend-Konfiguration.

Nachdem Astra Trident ein Backend erstellt oder aktualisiert hat, können Sie das Backend mit überprüfen `tridentctl` Oder das entsprechende `tridentbackend` CRD:

```
./tridentctl get backends ontap_nas_auto_export -n trident -o yaml
items:
- backendUUID: 403b5326-8482-40db-96d0-d83fb3f4daec
  config:
    aggregate: ""
    autoExportCIDRs:
    - 192.168.0.0/24
    autoExportPolicy: true
    backendName: ontap_nas_auto_export
    chapInitiatorSecret: ""
    chapTargetInitiatorSecret: ""
    chapTargetUsername: ""
    chapUsername: ""
    dataLIF: 192.168.0.135
    debug: false
    debugTraceFlags: null
    defaults:
      encryption: "false"
      exportPolicy: <automatic>
      fileType: ext4
```

Wenn Nodes zu einem Kubernetes-Cluster hinzugefügt und beim Astra Trident Controller registriert werden, werden die Exportrichtlinien vorhandener Back-Ends aktualisiert (vorausgesetzt, sie sind in den angegebenen Adressbereich enthalten `autoExportCIDRs` Für das Backend).

Wenn ein Node entfernt wird, überprüft Astra Trident alle Back-Ends, die online sind, um die Zugriffsregel für den Node zu entfernen. Indem Astra Trident diese Node-IP aus den Exportrichtlinien für gemanagte Back-Ends entfernt, verhindert er abnormale Mounts, sofern diese IP nicht von einem neuen Node im Cluster verwendet wird.

Aktualisieren Sie bei zuvor vorhandenen Back-Ends das Backend mit `tridentctl update backend` Stellt sicher, dass Astra Trident die Exportrichtlinien automatisch verwaltet. Dadurch wird eine neue Exportrichtlinie erstellt, die nach der UUID des Backends benannt ist und Volumes, die auf dem Backend vorhanden sind, verwenden die neu erstellte Exportrichtlinie, wenn sie wieder gemountet werden.



Wenn Sie ein Backend mit automatisch gemanagten Exportrichtlinien löschen, wird die dynamisch erstellte Exportrichtlinie gelöscht. Wenn das Backend neu erstellt wird, wird es als neues Backend behandelt und erzeugt eine neue Exportrichtlinie.

Wenn die IP-Adresse eines aktiven Node aktualisiert wird, müssen Sie den Astra Trident Pod auf dem Node neu starten. Astra Trident aktualisiert dann die Exportrichtlinie für Back-Ends, die es verwaltet, um diese IP-Änderung zu berücksichtigen.

Vorbereitung zur Bereitstellung von SMB Volumes

Mit ein wenig Vorbereitung können Sie SMB Volumes mit bereitstellen `ontap-nas` Treiber.



Zur Erstellung eines müssen Sie auf der SVM sowohl NFS- als auch SMB/CIFS-Protokolle konfigurieren `ontap-nas-economy` SMB Volume für ONTAP vor Ort: Ist eines dieser Protokolle nicht konfiguriert, schlägt die Erstellung von SMB Volumes fehl.

Bevor Sie beginnen

Bevor Sie SMB-Volumes bereitstellen können, müssen Sie über Folgendes verfügen:

- Kubernetes-Cluster mit einem Linux-Controller-Knoten und mindestens einem Windows-Worker-Node, auf dem Windows Server 2019 ausgeführt wird. Astra Trident unterstützt SMB Volumes, die nur auf Windows Nodes laufenden Pods gemountet werden.
- Mindestens ein Astra Trident-Geheimnis, der Ihre Active Directory-Anmeldedaten enthält. Um Geheimnis zu erzeugen `smbcreds`:

```
kubectl create secret generic smbcreds --from-literal username=user
--from-literal password='password'
```

- Ein CSI-Proxy, der als Windows-Dienst konfiguriert ist. Zum Konfigurieren von A `csi-proxy` Weitere Informationen finden Sie unter "[GitHub: CSI-Proxy](#)" Oder "[GitHub: CSI Proxy für Windows](#)" Für Kubernetes-Knoten, die auf Windows ausgeführt werden.

Schritte

1. Bei On-Premises-ONTAP können Sie optional eine SMB-Freigabe erstellen oder Astra Trident eine für Sie erstellen.



SMB-Freigaben sind für Amazon FSX for ONTAP erforderlich.

Sie können SMB-Admin-Freigaben auf zwei Arten erstellen: Mit "[Microsoft Management Console](#)" Snap-in für freigegebene Ordner oder mit der ONTAP-CLI. So erstellen Sie SMB-Freigaben mithilfe der ONTAP-CLI:

- a. Erstellen Sie bei Bedarf die Verzeichnispfadstruktur für die Freigabe.

Der `vserver cifs share create` Der Befehl überprüft während der Freigabenerstellung den in der Option `-path` angegebenen Pfad. Wenn der angegebene Pfad nicht vorhanden ist, schlägt der Befehl fehl.

- b. Erstellen einer mit der angegebenen SVM verknüpften SMB-Freigabe:

```
vserver cifs share create -vserver vserver_name -share-name
share_name -path path [-share-properties share_properties,...]
[other_attributes] [-comment text]
```

- c. Vergewissern Sie sich, dass die Freigabe erstellt wurde:

```
vserver cifs share show -share-name share_name
```



Siehe "[Erstellen Sie eine SMB-Freigabe](#)" Vollständige Informationen.

2. Beim Erstellen des Backend müssen Sie Folgendes konfigurieren, um SMB-Volumes festzulegen. Alle FSX-Konfigurationsoptionen für ONTAP-Backend finden Sie unter "[FSX für ONTAP Konfigurationsoptionen und Beispiele](#)".

| Parameter | Beschreibung | Beispiel |
|-----------------|---|---------------------------------|
| smbShare | <p>Sie können eine der folgenden Optionen angeben: Den Namen einer SMB-Freigabe, die mit der Microsoft Management Console oder der ONTAP-CLI erstellt wurde, einen Namen, über den Astra Trident die SMB-Freigabe erstellen kann, oder Sie können den Parameter leer lassen, um den Zugriff auf gemeinsame Freigaben auf Volumes zu verhindern.</p> <p>Dieser Parameter ist für On-Premises-ONTAP optional.</p> <p>Dieser Parameter ist für Amazon FSX for ONTAP-Back-Ends erforderlich und darf nicht leer sein.</p> | smb-share |
| nasType | Muss auf eingestellt sein smb. Wenn Null, wird standardmäßig auf gesetzt <code>nfs</code> . | smb |
| securityStyle | Sicherheitstyp für neue Volumes. Muss auf eingestellt sein ntfs Oder mixed Für SMB Volumes. | ntfs Oder mixed Für SMB Volumes |
| unixPermissions | Modus für neue Volumes. Muss für SMB Volumes leer gelassen werden. | “ ” |

ONTAP-NAS-Konfigurationsoptionen und Beispiele

Lernen Sie, wie Sie ONTAP NAS-Treiber mit Ihrer Astra Trident Installation erstellen und verwenden. Dieser Abschnitt enthält Beispiele und Details zur Back-End-Konfiguration für die Zuordnung von Back-Ends zu StorageClasses.

Back-End-Konfigurationsoptionen

Die Back-End-Konfigurationsoptionen finden Sie in der folgenden Tabelle:

| Parameter | Beschreibung | Standard |
|-------------------|---|---|
| version | | Immer 1 |
| storageDriverName | Name des Speichertreibers | „ontap-nas“, „ontap-nas-Economy“, „ontap-nas-flexgroup“, „ontap-san“, „ontap-san-Economy“ |
| backendName | Benutzerdefinierter Name oder das Storage-Backend | Treibername + „_“ + DatenLIF |

| Parameter | Beschreibung | Standard |
|------------------|--|---|
| managementLIF | <p>IP-Adresse eines Clusters oder einer SVM-Management-LIF</p> <p>Es kann ein vollständig qualifizierter Domänenname (FQDN) angegeben werden.</p> <p>Kann so eingestellt werden, dass IPv6-Adressen verwendet werden, wenn Astra Trident mit dem IPv6-Flag installiert wurde. IPv6-Adressen müssen in eckigen Klammern definiert werden, z. B. [28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555].</p> <p>Informationen zur nahtlosen MetroCluster-Umschaltung finden Sie im Beispiel: MetroCluster.</p> | „10.0.0.1“, „[2001:1234:abcd::fefe]“ |
| dataLIF | <p>IP-Adresse des LIF-Protokolls.</p> <p>Wir empfehlen Ihnen, anzugeben dataLIF. Falls nicht vorgesehen, ruft Astra Trident Daten-LIFs von der SVM ab. Sie können einen vollständig qualifizierten Domännennamen (FQDN) angeben, der für die NFS-Mount-Vorgänge verwendet werden soll. Damit können Sie ein Round-Robin-DNS zum Load-Balancing über mehrere Daten-LIFs erstellen.</p> <p>Kann nach der Anfangseinstellung geändert werden. Siehe .</p> <p>Kann so eingestellt werden, dass IPv6-Adressen verwendet werden, wenn Astra Trident mit dem IPv6-Flag installiert wurde. IPv6-Adressen müssen in eckigen Klammern definiert werden, z. B. [28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555].</p> <p>Für MetroCluster weglassen. Siehe Beispiel: MetroCluster.</p> | Angegebene Adresse oder abgeleitet von SVM, falls nicht angegeben (nicht empfohlen) |
| svm | <p>Zu verwendende Storage Virtual Machine</p> <p>Für MetroCluster weglassen. Siehe Beispiel: MetroCluster.</p> | Abgeleitet wenn eine SVM managementLIF Angegeben ist |
| autoExportPolicy | <p>Aktivieren Sie die automatische Erstellung von Exportrichtlinien und aktualisieren Sie [Boolean].</p> <p>Verwenden der autoExportPolicy Und autoExportCIDRs Optionen: Astra Trident kann Exportrichtlinien automatisch verwalten.</p> | Falsch |

| Parameter | Beschreibung | Standard |
|----------------------|--|--------------------------------------|
| autoExportCIDRs | Liste der CIDRs, nach denen die Node-IPs von Kubernetes gefiltert werden sollen autoExportPolicy Ist aktiviert. Verwenden der autoExportPolicy Und autoExportCIDRs Optionen: Astra Trident kann Exportrichtlinien automatisch verwalten. | [„0.0.0.0/0“, „:/0“] |
| labels | Satz willkürlicher JSON-formatierter Etiketten für Volumes | “ |
| clientCertificate | Base64-codierter Wert des Clientzertifikats. Wird für zertifikatbasierte Authentifizierung verwendet | “ |
| clientPrivateKey | Base64-kodierte Wert des privaten Client-Schlüssels. Wird für zertifikatbasierte Authentifizierung verwendet | “ |
| trustedCACertificate | Base64-kodierte Wert des vertrauenswürdigen CA-Zertifikats. Optional Wird für zertifikatbasierte Authentifizierung verwendet | “ |
| username | Benutzername für die Verbindung mit dem Cluster/SVM. Wird für Anmeldeinformationsbasierte verwendet | |
| password | Passwort für die Verbindung mit dem Cluster/SVM Wird für Anmeldeinformationsbasierte verwendet | |
| storagePrefix | Das Präfix wird beim Bereitstellen neuer Volumes in der SVM verwendet. Kann nicht aktualisiert werden, nachdem Sie sie festgelegt haben | trident |
| limitAggregateUsage | Bereitstellung fehlgeschlagen, wenn die Nutzung über diesem Prozentsatz liegt. Gilt nicht für Amazon FSX für ONTAP | „ (nicht standardmäßig durchgesetzt) |
| limitVolumeSize | Bereitstellung fehlgeschlagen, wenn die angeforderte Volume-Größe über diesem Wert liegt. Schränkt auch die maximale Größe der Volumes ein, die es für qtrees und LUNs verwaltet, und auf ein qtreesPerFlexvol Mit Option kann die maximale Anzahl von qtrees pro FlexVol angepasst werden. | „ (standardmäßig nicht erzwungen) |
| lunsPerFlexvol | Die maximale Anzahl an LUNs pro FlexVol muss im Bereich [50, 200] liegen. | „100“ |
| debugTraceFlags | Fehler-Flags bei der Fehlerbehebung beheben. Beispiel, {„API“:false, „method“:true} Verwenden Sie es nicht debugTraceFlags Es sei denn, Sie beheben Fehler und benötigen einen detaillierten Log Dump. | Null |

| Parameter | Beschreibung | Standard |
|------------------|--|-----------|
| nasType | <p>Konfiguration der Erstellung von NFS- oder SMB-Volumes</p> <p>Die Optionen lauten <code>nfs</code>, <code>smb</code> Oder null. Einstellung auf null setzt standardmäßig auf NFS-Volumes.</p> | nfs |
| nfsMountOptions | <p>Kommagetrennte Liste von NFS-Mount-Optionen.</p> <p>Die Mount-Optionen für Kubernetes-persistente Volumes werden normalerweise in Storage-Klassen angegeben. Wenn jedoch keine Mount-Optionen in einer Storage-Klasse angegeben sind, stellt Astra Trident die Mount-Optionen bereit, die in der Konfigurationsdatei des Storage-Back-End angegeben sind.</p> <p>Wenn in der Storage-Klasse oder der Konfigurationsdatei keine Mount-Optionen angegeben sind, stellt Astra Trident keine Mount-Optionen für ein damit verbundener persistentes Volume fest.</p> | “ |
| qtreesPerFlexvol | Maximale Ques pro FlexVol, muss im Bereich [50, 300] liegen | „200“ |
| smbShare | <p>Sie können eine der folgenden Optionen angeben: Den Namen einer SMB-Freigabe, die mit der Microsoft Management Console oder der ONTAP-CLI erstellt wurde, einen Namen, über den Astra Trident die SMB-Freigabe erstellen kann, oder Sie können den Parameter leer lassen, um den Zugriff auf gemeinsame Freigaben auf Volumes zu verhindern.</p> <p>Dieser Parameter ist für On-Premises-ONTAP optional.</p> <p>Dieser Parameter ist für Amazon FSX for ONTAP-Back-Ends erforderlich und darf nicht leer sein.</p> | smb-share |
| useREST | <p>Boolescher Parameter zur Verwendung von ONTAP REST-APIs. Technische Vorschau</p> <p><code>useREST</code> Wird als Tech-Vorschau bereitgestellt, das für Testumgebungen und nicht für Produktions-Workloads empfohlen wird. Wenn eingestellt auf <code>true</code>, Astra Trident wird ONTAP REST APIs zur Kommunikation mit dem Backend verwenden. Diese Funktion erfordert ONTAP 9.11.1 und höher. Darüber hinaus muss die verwendete ONTAP-Login-Rolle Zugriff auf den haben <code>ontap</code> Applikation. Dies wird durch die vordefinierte zufrieden <code>vsadmin</code> Und <code>cluster-admin</code> Rollen:</p> <p><code>useREST</code> Wird mit MetroCluster nicht unterstützt.</p> | Falsch |

Back-End-Konfigurationsoptionen für die Bereitstellung von Volumes

Sie können die Standardbereitstellung mit diesen Optionen im `defaults` Abschnitt der Konfiguration. Ein Beispiel finden Sie unten in den Konfigurationsbeispielen.

| Parameter | Beschreibung | Standard |
|--------------------------------|--|--|
| <code>spaceAllocation</code> | Speicherplatzzuweisung für LUNs | „Wahr“ |
| <code>spaceReserve</code> | Modus für Speicherplatzreservierung; „none“ (Thin) oder „Volume“ (Thick) | „Keine“ |
| <code>snapshotPolicy</code> | Die Snapshot-Richtlinie zu verwenden | „Keine“ |
| <code>qosPolicy</code> | QoS-Richtliniengruppe zur Zuweisung für erstellte Volumes Wählen Sie eine der <code>qosPolicy</code> oder <code>adaptiveQosPolicy</code> pro Storage Pool/Backend | “ |
| <code>adaptiveQosPolicy</code> | Adaptive QoS-Richtliniengruppe mit Zuordnung für erstellte Volumes Wählen Sie eine der <code>qosPolicy</code> oder <code>adaptiveQosPolicy</code> pro Storage Pool/Backend. Nicht unterstützt durch <code>ontap-nas</code> -Ökonomie | “ |
| <code>snapshotReserve</code> | Prozentsatz des für Snapshots reservierten Volumes | „0“ wenn <code>snapshotPolicy</code> Ist „keine“, andernfalls „“ |
| <code>splitOnClone</code> | Teilen Sie einen Klon bei der Erstellung von seinem übergeordneten Objekt auf | „Falsch“ |
| <code>encryption</code> | Aktivieren Sie NetApp Volume Encryption (NVE) auf dem neuen Volume, standardmäßig aktiviert <code>false</code> . NVE muss im Cluster lizenziert und aktiviert sein, damit diese Option verwendet werden kann. Wenn NAE auf dem Backend aktiviert ist, wird jedes im Astra Trident bereitgestellte Volume NAE aktiviert. Weitere Informationen finden Sie unter: " Astra Trident arbeitet mit NVE und NAE zusammen ". | „Falsch“ |
| <code>tieringPolicy</code> | Tiering-Richtlinie, die zu „keinen“ verwendet wird | „Nur snapshot“ für eine SVM-DR-Konfiguration vor ONTAP 9.5 |
| <code>unixPermissions</code> | Modus für neue Volumes | „777“ für NFS Volumes; leer (nicht zutreffend) für SMB Volumes |
| <code>snapshotDir</code> | Steuert den Zugriff auf das <code>.snapshot</code> Verzeichnis | „Falsch“ |
| <code>exportPolicy</code> | Zu verwendende Exportrichtlinie | „Standard“ |
| <code>securityStyle</code> | Sicherheitstyp für neue Volumes. NFS unterstützt <code>mixed</code> Und <code>unix</code> Sicherheitsstile. SMB-Support <code>mixed</code> Und <code>ntfs</code> Sicherheitsstile. | NFS-Standard ist <code>unix</code> . Der SMB-Standardwert ist <code>ntfs</code> . |



Die Verwendung von QoS Policy Groups mit Astra Trident erfordert ONTAP 9.8 oder höher. Es wird empfohlen, eine nicht gemeinsam genutzte QoS-Richtliniengruppe zu verwenden und sicherzustellen, dass die Richtliniengruppe auf jede Komponente einzeln angewendet wird. Eine Richtliniengruppe für Shared QoS führt zur Durchsetzung der Obergrenze für den Gesamtdurchsatz aller Workloads.

Beispiele für die Volume-Bereitstellung

Hier ein Beispiel mit definierten Standardwerten:

```
---
version: 1
storageDriverName: ontap-nas
backendName: customBackendName
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
labels:
  k8scluster: dev1
  backend: dev1-nasbackend
svm: trident_svm
username: cluster-admin
password: <password>
limitAggregateUsage: 80%
limitVolumeSize: 50Gi
nfsMountOptions: nfsvers=4
debugTraceFlags:
  api: false
  method: true
defaults:
  spaceReserve: volume
  qosPolicy: premium
  exportPolicy: myk8scluster
  snapshotPolicy: default
  snapshotReserve: '10'
```

Für `ontap-nas` und `ontap-nas-flexgroups` Astra Trident verwendet jetzt eine neue Berechnung, um sicherzustellen, dass die FlexVol korrekt mit dem Prozentwert der Snapshot Reserve und PVC dimensioniert ist. Wenn der Benutzer eine PVC anfordert, erstellt Astra Trident unter Verwendung der neuen Berechnung die ursprüngliche FlexVol mit mehr Speicherplatz. Diese Berechnung stellt sicher, dass der Benutzer den beschreibbaren Speicherplatz erhält, für den er in der PVC benötigt wird, und nicht weniger Speicherplatz als der angeforderte. Vor Version 2.07, wenn der Benutzer eine PVC anfordert (z. B. 5 gib), bei der SnapshotReserve auf 50 Prozent, erhalten sie nur 2,5 gib schreibbaren Speicherplatz. Der Grund dafür ist, dass der Benutzer das gesamte Volume und angefordert hat `snapshotReserve ist ein Prozentsatz davon. Mit Trident 21.07 sind die Benutzeranforderungen der beschreibbare Speicherplatz, und Astra Trident definiert den snapshotReserve Zahl als Prozentsatz des gesamten Volumens. Dies gilt`

nicht für `ontap-nas-economy`. Im folgenden Beispiel sehen Sie, wie das funktioniert:

Die Berechnung ist wie folgt:

```
Total volume size = (PVC requested size) / (1 - (snapshotReserve
percentage) / 100)
```

Für die `snapshotReserve = 50 %`, und die PVC-Anfrage = 5 gib, beträgt die Gesamtgröße des Volumes $2/5 = 10$ gib, und die verfügbare Größe beträgt 5 gib. Dies entspricht dem, was der Benutzer in der PVC-Anfrage angefordert hat. Der `volume show` Der Befehl sollte Ergebnisse anzeigen, die diesem Beispiel ähnlich sind:

| Vserver | Volume | Aggregate | State | Type | Size | Available | Used% |
|---------|--|-----------|--------|------|------|-----------|-------|
| | <code>_pvc_89f1c156_3801_4de4_9f9d_034d54c395f4</code> | | online | RW | 10GB | 5.00GB | 0% |
| | <code>_pvc_e8372153_9ad9_474a_951a_08ae15e1c0ba</code> | | online | RW | 1GB | 511.8MB | 0% |

2 entries were displayed.

Vorhandene Back-Ends aus vorherigen Installationen stellen Volumes wie oben beschrieben beim Upgrade von Astra Trident bereit. Bei Volumes, die Sie vor dem Upgrade erstellt haben, sollten Sie die Größe ihrer Volumes entsprechend der zu beobachtenden Änderung anpassen. Beispiel: Ein 2 gib PVC mit `snapshotReserve=50` Früher hat ein Volume ergeben, das 1 gib beschreibbaren Speicherplatz bereitstellt. Wenn Sie die Größe des Volumes auf 3 gib ändern, z. B. stellt die Applikation auf einem 6 gib an beschreibbarem Speicherplatz bereit.

Minimale Konfigurationsbeispiele

Die folgenden Beispiele zeigen grundlegende Konfigurationen, bei denen die meisten Parameter standardmäßig belassen werden. Dies ist der einfachste Weg, ein Backend zu definieren.



Wenn Sie Amazon FSX auf NetApp ONTAP mit Trident verwenden, empfiehlt es sich, DNS-Namen für LIFs anstelle von IP-Adressen anzugeben.

Beispiel für die NAS-Ökonomie von ONTAP

```
---
version: 1
storageDriverName: ontap-nas-economy
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
svm: svm_nfs
username: vsadmin
password: password
```

Beispiel für ONTAP NAS FlexGroup

```
---
version: 1
storageDriverName: ontap-nas-flexgroup
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
svm: svm_nfs
username: vsadmin
password: password
```

Beispiel: MetroCluster

Sie können das Backend so konfigurieren, dass die Backend-Definition nach Umschaltung und einem Wechsel während nicht manuell aktualisiert werden muss ["SVM-Replizierung und Recovery"](#).

Für nahtloses Switchover und Switchback geben Sie die SVM über an `managementLIF` Und lassen Sie die aus `dataLIF` Und `svm` Parameter. Beispiel:

```
---
version: 1
storageDriverName: ontap-nas
managementLIF: 192.168.1.66
username: vsadmin
password: password
```

Beispiel: SMB Volumes

```
---
version: 1
backendName: ExampleBackend
storageDriverName: ontap-nas
managementLIF: 10.0.0.1
nasType: smb
securityStyle: ntfs
unixPermissions: ""
dataLIF: 10.0.0.2
svm: svm_nfs
username: vsadmin
password: password
```


Beispiel für die zertifikatbasierte Authentifizierung

Dies ist ein minimales Beispiel für die Back-End-Konfiguration. `clientCertificate`, `clientPrivateKey`, und `trustedCACertificate` (Optional, wenn Sie eine vertrauenswürdige CA verwenden) werden ausgefüllt `backend.json` Und nehmen Sie die base64-kodierten Werte des Clientzertifikats, des privaten Schlüssels und des vertrauenswürdigen CA-Zertifikats.

```
---
version: 1
backendName: DefaultNASBackend
storageDriverName: ontap-nas
managementLIF: 10.0.0.1
dataLIF: 10.0.0.15
svm: nfs_svm
clientCertificate: ZXR0ZXJwYXB...ICMgJ3BhcGVyc2
clientPrivateKey: vciwKIyAgZG...0cnksIGRlc2NyaX
trustedCACertificate: zcyBbaG...b3Igb3duIGNsYXNz
storagePrefix: myPrefix_
```

Beispiel für eine Richtlinie für den automatischen Export

In diesem Beispiel erfahren Sie, wie Sie Astra Trident anweisen können, dynamische Exportrichtlinien zu verwenden, um die Exportrichtlinie automatisch zu erstellen und zu verwalten. Das funktioniert auch für das `ontap-nas-economy` Und `ontap-nas-flexgroup` Treiber.

```
---
version: 1
storageDriverName: ontap-nas
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
svm: svm_nfs
labels:
  k8scluster: test-cluster-east-1a
  backend: test1-nasbackend
autoExportPolicy: true
autoExportCIDRs:
- 10.0.0.0/24
username: admin
password: password
nfsMountOptions: nfsvers=4
```

Beispiel für IPv6-Adressen

Dieses Beispiel zeigt managementLIF Verwenden einer IPv6-Adresse.

```
---
version: 1
storageDriverName: ontap-nas
backendName: nas_ipv6_backend
managementLIF: "[5c5d:5edf:8f:7657:bef8:109b:1b41:d491]"
labels:
  k8scluster: test-cluster-east-1a
  backend: test1-ontap-ipv6
svm: nas_ipv6_svm
username: vsadmin
password: password
```

Amazon FSX für ONTAP mit SMB-Volumes – Beispiel

Der smbShare Parameter ist für FSX for ONTAP mit SMB Volumes erforderlich.

```
---
version: 1
backendName: SMBBackend
storageDriverName: ontap-nas
managementLIF: example.mgmt.fqdn.aws.com
nasType: smb
dataLIF: 10.0.0.15
svm: nfs_svm
smbShare: smb-share
clientCertificate: ZXR0ZXJwYXB...ICMgJ3BhcGVyc2
clientPrivateKey: vciwKIyAgZG...0cnksIGRlc2NyaX
trustedCACertificate: zcyBbaG...b3Igb3duIGNsYXNz
storagePrefix: myPrefix_
```

Beispiele für Back-Ends mit virtuellen Pools

In den unten gezeigten Beispieldateien für die Backend-Definition werden spezifische Standardwerte für alle Speicherpools festgelegt, z. B. spaceReserve Bei keiner, spaceAllocation Bei false, und encryption Bei false. Die virtuellen Pools werden im Abschnitt Speicher definiert.

Astra Trident bestimmt die Bereitstellungsetiketten im Feld „Kommentare“. Kommentare werden auf FlexVol für gesetzt ontap-nas Oder FlexGroup für ontap-nas-flexgroup. Astra Trident kopiert alle Labels auf einem virtuellen Pool auf das Storage-Volume während der Bereitstellung. Storage-Administratoren können Labels je virtuellen Pool definieren und Volumes nach Label gruppieren.

In diesen Beispielen legen einige Speicherpools eigene fest `spaceReserve`, `spaceAllocation`, und `encryption` Werte und einige Pools überschreiben die Standardwerte.

Beispiel: ONTAP NAS

```
---
version: 1
storageDriverName: ontap-nas
managementLIF: 10.0.0.1
svm: svm_nfs
username: admin
password: <password>
nfsMountOptions: nfsvers=4
defaults:
  spaceReserve: none
  encryption: 'false'
  qosPolicy: standard
labels:
  store: nas_store
  k8scluster: prod-cluster-1
region: us_east_1
storage:
- labels:
  app: msoffice
  cost: '100'
  zone: us_east_1a
  defaults:
    spaceReserve: volume
    encryption: 'true'
    unixPermissions: '0755'
    adaptiveQosPolicy: adaptive-premium
- labels:
  app: slack
  cost: '75'
  zone: us_east_1b
  defaults:
    spaceReserve: none
    encryption: 'true'
    unixPermissions: '0755'
- labels:
  department: legal
  creditpoints: '5000'
  zone: us_east_1b
  defaults:
    spaceReserve: none
    encryption: 'true'
    unixPermissions: '0755'
- labels:
  app: wordpress
```

```
    cost: '50'  
    zone: us_east_1c  
    defaults:  
      spaceReserve: none  
      encryption: 'true'  
      unixPermissions: '0775'  
- labels:  
  app: mysqldb  
  cost: '25'  
  zone: us_east_1d  
  defaults:  
    spaceReserve: volume  
    encryption: 'false'  
    unixPermissions: '0775'
```

Beispiel für ONTAP NAS FlexGroup

```
---
version: 1
storageDriverName: ontap-nas-flexgroup
managementLIF: 10.0.0.1
svm: svm_nfs
username: vsadmin
password: <password>
defaults:
  spaceReserve: none
  encryption: 'false'
labels:
  store: flexgroup_store
  k8scluster: prod-cluster-1
region: us_east_1
storage:
- labels:
  protection: gold
  creditpoints: '50000'
  zone: us_east_1a
  defaults:
    spaceReserve: volume
    encryption: 'true'
    unixPermissions: '0755'
- labels:
  protection: gold
  creditpoints: '30000'
  zone: us_east_1b
  defaults:
    spaceReserve: none
    encryption: 'true'
    unixPermissions: '0755'
- labels:
  protection: silver
  creditpoints: '20000'
  zone: us_east_1c
  defaults:
    spaceReserve: none
    encryption: 'true'
    unixPermissions: '0775'
- labels:
  protection: bronze
  creditpoints: '10000'
  zone: us_east_1d
  defaults:
```

```
spaceReserve: volume  
encryption: 'false'  
unixPermissions: '0775'
```

Beispiel für die NAS-Ökonomie von ONTAP

```
---
version: 1
storageDriverName: ontap-nas-economy
managementLIF: 10.0.0.1
svm: svm_nfs
username: vsadmin
password: <password>
defaults:
  spaceReserve: none
  encryption: 'false'
labels:
  store: nas_economy_store
region: us_east_1
storage:
- labels:
  department: finance
  creditpoints: '6000'
  zone: us_east_1a
  defaults:
    spaceReserve: volume
    encryption: 'true'
    unixPermissions: '0755'
- labels:
  protection: bronze
  creditpoints: '5000'
  zone: us_east_1b
  defaults:
    spaceReserve: none
    encryption: 'true'
    unixPermissions: '0755'
- labels:
  department: engineering
  creditpoints: '3000'
  zone: us_east_1c
  defaults:
    spaceReserve: none
    encryption: 'true'
    unixPermissions: '0775'
- labels:
  department: humanresource
  creditpoints: '2000'
  zone: us_east_1d
  defaults:
    spaceReserve: volume
```



```
encryption: 'false'  
unixPermissions: '0775'
```

Back-Ends StorageClasses zuordnen

Die folgenden StorageClass-Definitionen finden Sie unter [Beispiele für Back-Ends mit virtuellen Pools](#).

Verwenden der `parameters.selector` Jede StorageClass ruft auf, welche virtuellen Pools zum Hosten eines Volumes verwendet werden können. Auf dem Volume werden die Aspekte im ausgewählten virtuellen Pool definiert.

- Der `protection-gold` StorageClass wird dem ersten und zweiten virtuellen Pool in zugeordnet `ontap-nas-flexgroup` Back-End: Dies sind die einzigen Pools, die Gold-Level-Schutz bieten.

```
apiVersion: storage.k8s.io/v1  
kind: StorageClass  
metadata:  
  name: protection-gold  
provisioner: netapp.io/trident  
parameters:  
  selector: "protection=gold"  
  fsType: "ext4"
```

- Der `protection-not-gold` StorageClass wird dem dritten und vierten virtuellen Pool in zugeordnet `ontap-nas-flexgroup` Back-End: Dies sind die einzigen Pools, die Schutz Level nicht Gold bieten.

```
apiVersion: storage.k8s.io/v1  
kind: StorageClass  
metadata:  
  name: protection-not-gold  
provisioner: netapp.io/trident  
parameters:  
  selector: "protection!=gold"  
  fsType: "ext4"
```

- Der `app-mysqldb` StorageClass wird dem vierten virtuellen Pool in zugeordnet `ontap-nas` Back-End: Dies ist der einzige Pool, der Storage-Pool-Konfiguration für `mysqldb`-Typ-App bietet.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: app-mysqldb
provisioner: netapp.io/trident
parameters:
  selector: "app=mysqldb"
  fsType: "ext4"
```

- The `protection-silver-creditpoints-20k` StorageClass wird dem dritten virtuellen Pool in zugeordnet `ontap-nas-flexgroup` Back-End: Dies ist der einzige Pool mit Silber-Level-Schutz und 20000 Kreditpunkte.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-silver-creditpoints-20k
provisioner: netapp.io/trident
parameters:
  selector: "protection=silver; creditpoints=20000"
  fsType: "ext4"
```

- Der `creditpoints-5k` StorageClass wird dem dritten virtuellen Pool in zugeordnet `ontap-nas` Back-End und der zweite virtuelle Pool im `ontap-nas-economy` Back-End: Dies sind die einzigen Poolangebote mit 5000 Kreditpunkten.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: creditpoints-5k
provisioner: netapp.io/trident
parameters:
  selector: "creditpoints=5000"
  fsType: "ext4"
```

Astra Trident entscheidet, welcher virtuelle Pool ausgewählt wird und stellt sicher, dass die Storage-Anforderungen erfüllt werden.

Aktualisierung `dataLIF` Nach der Erstkonfiguration

Sie können die Daten-LIF nach der Erstkonfiguration ändern, indem Sie den folgenden Befehl ausführen, um die neue Backend-JSON-Datei mit aktualisierten Daten-LIF bereitzustellen.

```
tridentctl update backend <backend-name> -f <path-to-backend-json-file-with-updated-dataLIF>
```



Wenn PVCs an einen oder mehrere Pods angeschlossen sind, müssen Sie alle entsprechenden Pods herunterfahren und sie dann wieder zurückbringen, damit die neue logische Daten wirksam werden.

Amazon FSX für NetApp ONTAP

Setzen Sie Astra Trident mit Amazon FSX für NetApp ONTAP ein

"Amazon FSX für NetApp ONTAP" ist ein vollständig gemanagter AWS Service, mit dem Kunden Filesysteme auf Basis des NetApp ONTAP Storage-Betriebssystems starten und ausführen können. Mit FSX für ONTAP können Sie bekannte NetApp Funktionen sowie die Performance und Administration nutzen und gleichzeitig die Einfachheit, Agilität, Sicherheit und Skalierbarkeit beim Speichern von Daten in AWS nutzen. FSX für ONTAP unterstützt ONTAP Dateisystemfunktionen und Administrations-APIs.

Überblick

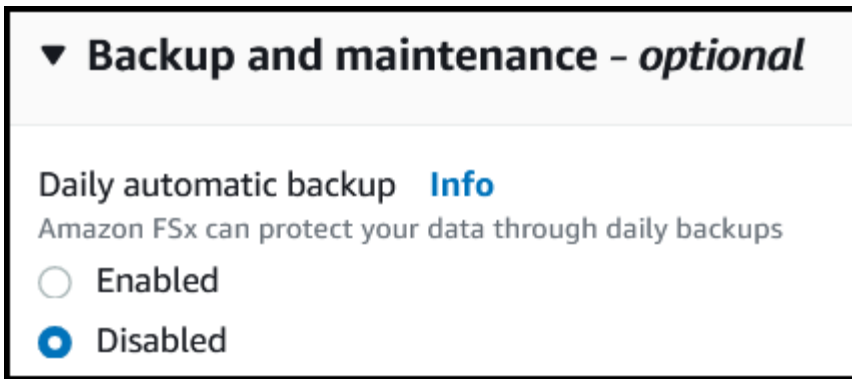
Ein Dateisystem ist die primäre Ressource in Amazon FSX, analog zu einem ONTAP-Cluster vor Ort. Innerhalb jeder SVM können Sie ein oder mehrere Volumes erstellen, bei denen es sich um Daten-Container handelt, die die Dateien und Ordner im Filesystem speichern. Amazon FSX für NetApp ONTAP wird Data ONTAP als gemanagtes Dateisystem in der Cloud zur Verfügung stellen. Der neue Dateisystemtyp heißt **NetApp ONTAP**.

Mit Astra Trident mit Amazon FSX für NetApp ONTAP können Sie sicherstellen, dass Kubernetes Cluster, die in Amazon Elastic Kubernetes Service (EKS) ausgeführt werden, persistente Block- und Datei-Volumes bereitstellen, die durch ONTAP gesichert sind.

Amazon FSX für NetApp ONTAP "[FabricPool](#)" Für das Management von Storage Tiers benötigen. Diese ermöglicht die Speicherung von Daten in einer Tier, basierend darauf, ob häufig auf die Daten zugegriffen wird.

Überlegungen

- SMB Volumes:
 - SMB Volumes werden mit unterstützt `ontap-nas` Nur Treiber.
 - Astra Trident unterstützt SMB Volumes, die nur auf Windows Nodes laufenden Pods gemountet werden.
- Volumes, die auf Amazon FSX Filesystemen erstellt wurden und bei denen automatische Backups aktiviert sind, können nicht durch Trident gelöscht werden. Um PVCs zu löschen, müssen Sie das PV und das FSX für ONTAP-Volume manuell löschen. So vermeiden Sie dieses Problem:
 - Verwenden Sie nicht **Quick create**, um das FSX für das ONTAP-Dateisystem zu erstellen. Der Quick-Create-Workflow ermöglicht automatische Backups und bietet keine Opt-out-Option.
 - Bei Verwendung von **Standard create** deaktivieren Sie die automatische Sicherung. Durch Deaktivieren automatischer Backups kann Trident ein Volume erfolgreich ohne weitere manuelle Eingriffe löschen.



FSX für ONTAP-Treiber Details

Sie können Astra Trident mithilfe der folgenden Treiber in Amazon FSX für NetApp ONTAP integrieren:

- `ontap-san`: Jedes bereitgestellte PV ist eine LUN innerhalb seines eigenen Amazon FSX für NetApp ONTAP Volume.
- `ontap-san-economy`: Jedes bereitgestellte PV ist eine LUN mit einer konfigurierbaren Anzahl an LUNs pro Amazon FSX für das NetApp ONTAP Volume.
- `ontap-nas`: Jedes bereitgestellte PV ist ein vollständiger Amazon FSX für NetApp ONTAP Volume.
- `ontap-nas-economy`: Jedes bereitgestellte PV ist ein qtree mit einer konfigurierbaren Anzahl von qtrees pro Amazon FSX für NetApp ONTAP Volume.
- `ontap-nas-flexgroup`: Jedes bereitgestellte PV ist ein vollständiger Amazon FSX für NetApp ONTAP FlexGroup Volume.

Informationen zu den Fahrern finden Sie unter "[NAS-Treiber](#)" Und "[SAN-Treiber](#)".

Authentifizierung

Astra Trident bietet zwei Authentifizierungsmodi.

- **Zertifikatsbasiert**: Astra Trident kommuniziert mit der SVM auf Ihrem FSX Dateisystem mit einem Zertifikat, das auf Ihrer SVM installiert ist.
- **Anmeldeinformationsbasiert**: Sie können den verwenden `fsxadmin` Benutzer für Ihr Dateisystem oder die `vsadmin` Benutzer für Ihre SVM konfiguriert.



Astra Trident erwartet einen weiteren Betrieb `vsadmin` SVM-Benutzer oder als Benutzer mit einem anderen Namen, der dieselbe Rolle hat. Amazon FSX für NetApp ONTAP hat eine `fsxadmin` Benutzer, die nur einen eingeschränkten Ersatz für die ONTAP bieten `admin` Cluster-Benutzer. Wir empfehlen Ihnen sehr, es zu verwenden `vsadmin` Mit Astra Trident:

Sie können Back-Ends aktualisieren, um zwischen auf Anmeldeinformationen basierenden und zertifikatbasierten Methoden zu verschieben. Wenn Sie jedoch versuchen, **Anmeldeinformationen und Zertifikate** bereitzustellen, schlägt die Backend-Erstellung fehl. Um zu einer anderen Authentifizierungsmethode zu wechseln, müssen Sie die vorhandene Methode von der Backend-Konfiguration entfernen.

Weitere Informationen zur Aktivierung der Authentifizierung finden Sie in der Authentifizierung für Ihren Treibertyp:

- ["ONTAP NAS-Authentifizierung"](#)
- ["ONTAP SAN-Authentifizierung"](#)

Weitere Informationen

- ["Dokumentation zu Amazon FSX für NetApp ONTAP"](#)
- ["Blogbeitrag zu Amazon FSX für NetApp ONTAP"](#)

Integration von Amazon FSX für NetApp ONTAP

Sie können Ihr Filesystem Amazon FSX für NetApp ONTAP mit Astra Trident integrieren, um sicherzustellen, dass Kubernetes Cluster, die in Amazon Elastic Kubernetes Service (EKS) ausgeführt werden, persistente Block- und File-Volumes mit ONTAP bereitstellen können.

Anforderungen

Zusätzlich zu ["Anforderungen von Astra Trident"](#) Zur Integration von FSX für ONTAP mit Astra Trident benötigen Sie Folgendes:

- Ein vorhandener Amazon EKS-Cluster oder selbst verwalteter Kubernetes-Cluster mit `kubectl` installiert.
- Ein vorhandenes Amazon FSX for NetApp ONTAP-Filesystem und eine Storage Virtual Machine (SVM), die über die Worker-Nodes Ihres Clusters erreichbar ist.
- Worker-Nodes, die vorbereitet sind ["NFS oder iSCSI"](#).



Achten Sie darauf, dass Sie die für Amazon Linux und Ubuntu erforderlichen Schritte zur Knotenvorbereitung befolgen ["Amazon Machine Images"](#) (Amis) je nach EKS AMI-Typ.

- Astra Trident unterstützt SMB Volumes, die nur auf Windows Nodes laufenden Pods gemountet werden. Siehe [Vorbereitung zur Bereitstellung von SMB Volumes](#) Entsprechende Details.

Integration von ONTAP-SAN- und NAS-Treibern



Wenn Sie für SMB Volumes konfigurieren, müssen Sie lesen [Vorbereitung zur Bereitstellung von SMB Volumes](#) Bevor Sie das Backend erstellen.

Schritte

1. Implementieren Sie Astra Trident mit einer der Lösungen ["Implementierungsoptionen"](#).
2. Sammeln Sie den SVM-Management-LIF-DNS-Namen. Suchen Sie zum Beispiel mit der AWS CLI nach DNSName Eintrag unter `Endpoints` → `Management` Nach Ausführung des folgenden Befehls:

```
aws fsx describe-storage-virtual-machines --region <file system region>
```

3. Erstellen und Installieren von Zertifikaten für ["NAS-Back-End-Authentifizierung"](#) Oder ["SAN-Back-End-Authentifizierung"](#).



Sie können sich bei Ihrem Dateisystem anmelden (zum Beispiel Zertifikate installieren) mit SSH von überall, wo Sie Ihr Dateisystem erreichen können. Verwenden Sie die `fsxadmin` Benutzer, das Kennwort, das Sie beim Erstellen Ihres Dateisystems konfiguriert haben, und der Management-DNS-Name von `aws fsx describe-file-systems`.

4. Erstellen Sie eine Backend-Datei mithilfe Ihrer Zertifikate und des DNS-Namens Ihrer Management LIF, wie im folgenden Beispiel dargestellt:

YAML

```
---
version: 1
storageDriverName: ontap-san
backendName: customBackendName
managementLIF: svm-XXXXXXXXXXXXXXXXXX.fs-XXXXXXXXXXXXXXXXXXXX.fsx.us-
east-2.aws.internal
svm: svm01
clientCertificate: ZXR0ZXJwYXB...ICMgJ3BhcGVyc2
clientPrivateKey: vciwKIyAgZG...0cnksIGRlc2NyaX
trustedCACertificate: zcyBbaG...b3Igb3duIGNsYXNz
```

JSON

```
{
  "version": 1,
  "storageDriverName": "ontap-san",
  "backendName": "customBackendName",
  "managementLIF": "svm-XXXXXXXXXXXXXXXXXXXX.fs-
XXXXXXXXXXXXXXXXXXXX.fsx.us-east-2.aws.internal",
  "svm": "svm01",
  "clientCertificate": "ZXR0ZXJwYXB...ICMgJ3BhcGVyc2",
  "clientPrivateKey": "vciwKIyAgZG...0cnksIGRlc2NyaX",
  "trustedCACertificate": "zcyBbaG...b3Igb3duIGNsYXNz"
}
```

Informationen zum Erstellen von Back-Ends finden Sie unter folgenden Links:

- ["Konfigurieren Sie ein Backend mit ONTAP NAS-Treibern"](#)
- ["Konfigurieren Sie ein Backend mit ONTAP-SAN-Treibern"](#)

Vorbereitung zur Bereitstellung von SMB Volumes

Sie können SMB-Volumes mit bereitstellen `ontap-nas` Treiber. Bevor Sie fertig sind [Integration von ONTAP-SAN- und NAS-Treibern](#) Führen Sie die folgenden Schritte aus.

Bevor Sie beginnen

Bevor Sie SMB-Volumes mit bereitstellen können `ontap-nas` Treiber, müssen Sie Folgendes haben.

- Kubernetes-Cluster mit einem Linux-Controller-Knoten und mindestens einem Windows-Worker-Node, auf dem Windows Server 2019 ausgeführt wird. Astra Trident unterstützt SMB Volumes, die nur auf Windows Nodes laufenden Pods gemountet werden.
- Mindestens ein Astra Trident-Geheimnis, der Ihre Active Directory-Anmeldedaten enthält. Um Geheimnis zu erzeugen `smbcreds`:

```
kubectl create secret generic smbcreds --from-literal username=user
--from-literal password='password'
```

- Ein CSI-Proxy, der als Windows-Dienst konfiguriert ist. Zum Konfigurieren von A `csi-proxy``Weitere Informationen finden Sie unter "[GitHub: CSI-Proxy](#)" Oder "[GitHub: CSI Proxy für Windows](#)" Für Kubernetes-Knoten, die auf Windows ausgeführt werden.

Schritte

1. Erstellen von SMB-Freigaben Sie können SMB-Admin-Freigaben auf zwei Arten erstellen: Mit "[Microsoft Management Console](#)" Snap-in für freigegebene Ordner oder mit der ONTAP-CLI. So erstellen Sie SMB-Freigaben mithilfe der ONTAP-CLI:

- a. Erstellen Sie bei Bedarf die Verzeichnispfadstruktur für die Freigabe.

Der `vserver cifs share create` Der Befehl überprüft während der Freigabenerstellung den in der Option `-path` angegebenen Pfad. Wenn der angegebene Pfad nicht vorhanden ist, schlägt der Befehl fehl.

- b. Erstellen einer mit der angegebenen SVM verknüpften SMB-Freigabe:

```
vserver cifs share create -vserver vserver_name -share-name
share_name -path path [-share-properties share_properties,...]
[other_attributes] [-comment text]
```

- c. Vergewissern Sie sich, dass die Freigabe erstellt wurde:

```
vserver cifs share show -share-name share_name
```



Siehe "[Erstellen Sie eine SMB-Freigabe](#)" Vollständige Informationen.

2. Beim Erstellen des Backend müssen Sie Folgendes konfigurieren, um SMB-Volumes festzulegen. Alle FSX-Konfigurationsoptionen für ONTAP-Backend finden Sie unter "[FSX für ONTAP Konfigurationsoptionen und Beispiele](#)".

| Parameter | Beschreibung | Beispiel |
|-----------------|---|---------------------------------|
| smbShare | Sie können eine der folgenden Optionen angeben: Den Namen einer SMB-Freigabe, die mit der Microsoft Management Console oder der ONTAP-CLI erstellt wurde, oder einen Namen, mit dem Astra Trident die SMB-Freigabe erstellen kann. Dieser Parameter ist für Amazon FSX for ONTAP Back-Ends erforderlich. | smb-share |
| nasType | Muss auf eingestellt sein smb. Wenn Null, wird standardmäßig auf gesetzt nfs. | smb |
| securityStyle | Sicherheitstyp für neue Volumes. Muss auf eingestellt sein ntfs Oder mixed Für SMB Volumes. | ntfs Oder mixed Für SMB Volumes |
| unixPermissions | Modus für neue Volumes. Muss für SMB Volumes leer gelassen werden. | “ ” |

FSX für ONTAP Konfigurationsoptionen und Beispiele

Erfahren Sie mehr über Back-End-Konfigurationsoptionen für Amazon FSX für ONTAP. Dieser Abschnitt enthält Beispiele für die Back-End-Konfiguration.

Back-End-Konfigurationsoptionen

Die Back-End-Konfigurationsoptionen finden Sie in der folgenden Tabelle:

| Parameter | Beschreibung | Beispiel |
|-------------------|---|---|
| version | | Immer 1 |
| storageDriverName | Name des Speichertreibers | ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, ontap-san-economy |
| backendName | Benutzerdefinierter Name oder das Storage-Backend | Treibername + „_“ + DatenLIF |

| Parameter | Beschreibung | Beispiel |
|---------------|--|--------------------------------------|
| managementLIF | <p>IP-Adresse eines Clusters oder einer SVM-Management-LIF</p> <p>Es kann ein vollständig qualifizierter Domänenname (FQDN) angegeben werden.</p> <p>Kann so eingestellt werden, dass IPv6-Adressen verwendet werden, wenn Astra Trident mit dem IPv6-Flag installiert wurde. IPv6-Adressen müssen in eckigen Klammern definiert werden, z. B. [28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555].</p> | „10.0.0.1“, „[2001:1234:abcd::fefe]“ |
| dataLIF | <p>IP-Adresse des LIF-Protokolls.</p> <p>ONTAP NAS drivers: Wir empfehlen die Angabe von dataLIF. Falls nicht vorgesehen, ruft Astra Trident Daten-LIFs von der SVM ab. Sie können einen vollständig qualifizierten Domännennamen (FQDN) angeben, der für die NFS-Mount-Vorgänge verwendet werden soll. Damit können Sie ein Round-Robin-DNS zum Load-Balancing über mehrere Daten-LIFs erstellen. Kann nach der Anfangseinstellung geändert werden. Siehe .</p> <p>ONTAP-SAN-Treiber: Geben Sie nicht für iSCSI an. Astra Trident verwendet die ONTAP Selective LUN Map, um die iSCI LIFs zu ermitteln, die für die Einrichtung einer Multi-Path-Sitzung erforderlich sind. Eine Warnung wird erzeugt, wenn dataLIF explizit definiert ist.</p> <p>Kann so eingestellt werden, dass IPv6-Adressen verwendet werden, wenn Astra Trident mit dem IPv6-Flag installiert wurde. IPv6-Adressen müssen in eckigen Klammern definiert werden, z. B. [28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555].</p> | |

| Parameter | Beschreibung | Beispiel |
|----------------------|--|---|
| autoExportPolicy | Aktivieren Sie die automatische Erstellung von Exportrichtlinien und aktualisieren Sie [Boolean]. Verwenden der autoExportPolicy Und autoExportCIDRs Optionen: Astra Trident kann Exportrichtlinien automatisch verwalten. | false |
| autoExportCIDRs | Liste der CIDRs, nach denen die Node-IPs von Kubernetes gefiltert werden sollen autoExportPolicy Ist aktiviert. Verwenden der autoExportPolicy Und autoExportCIDRs Optionen: Astra Trident kann Exportrichtlinien automatisch verwalten. | „[„0.0.0.0/0“, „:/0“]“ |
| labels | Satz willkürlicher JSON-formatierter Etiketten für Volumes | “ |
| clientCertificate | Base64-codierter Wert des Clientzertifikats. Wird für zertifikatbasierte Authentifizierung verwendet | “ |
| clientPrivateKey | Base64-kodierte Wert des privaten Client-Schlüssels. Wird für zertifikatbasierte Authentifizierung verwendet | “ |
| trustedCACertificate | Base64-kodierte Wert des vertrauenswürdigen CA-Zertifikats. Optional Wird für die zertifikatbasierte Authentifizierung verwendet. | “ |
| username | Benutzername zum Herstellen einer Verbindung zum Cluster oder zur SVM. Wird für die Anmeldeinformationsbasierte Authentifizierung verwendet. Beispiel: Vsadmin. | |
| password | Passwort für die Verbindung mit dem Cluster oder der SVM Wird für die Anmeldeinformationsbasierte Authentifizierung verwendet. | |
| svm | Zu verwendende Storage Virtual Machine | Abgeleitet, wenn eine SVM Management LIF angegeben ist. |

| Parameter | Beschreibung | Beispiel |
|---------------------|--|---------------------------------------|
| storagePrefix | <p>Das Präfix wird beim Bereitstellen neuer Volumes in der SVM verwendet.</p> <p>Kann nach der Erstellung nicht geändert werden. Um diesen Parameter zu aktualisieren, müssen Sie ein neues Backend erstellen.</p> | trident |
| limitAggregateUsage | <p>Nicht für Amazon FSX für NetApp ONTAP angeben.</p> <p>Die vorhanden <code>fsxadmin</code> Und <code>vsadmin</code> Enthalten Sie nicht die erforderlichen Berechtigungen, um die Aggregatnutzung abzurufen und sie mit Astra Trident zu begrenzen.</p> | Verwenden Sie ihn nicht. |
| limitVolumeSize | <p>Bereitstellung fehlgeschlagen, wenn die angeforderte Volume-Größe über diesem Wert liegt.</p> <p>Schränkt auch die maximale Größe der Volumes ein, die es für qtrees und LUNs verwaltet, und auf ein <code>qtreesPerFlexvol</code> Mit Option kann die maximale Anzahl von qtrees pro FlexVol angepasst werden.</p> | „“ (nicht standardmäßig durchgesetzt) |
| lunsPerFlexvol | <p>Die maximale Anzahl an LUNs pro FlexVol muss im Bereich [50, 200] liegen.</p> <p>Nur SAN</p> | 100 |
| debugTraceFlags | <p>Fehler-Flags bei der Fehlerbehebung beheben. Beispiel: { „API“:false, „Methode“:true}</p> <p>Verwenden Sie es nicht <code>debugTraceFlags</code> Es sei denn, Sie beheben Fehler und benötigen einen detaillierten Log Dump.</p> | Null |

| Parameter | Beschreibung | Beispiel |
|------------------|--|-----------|
| nfsMountOptions | <p>Kommagetrennte Liste von NFS-Mount-Optionen.</p> <p>Die Mount-Optionen für Kubernetes-persistente Volumes werden normalerweise in Storage-Klassen angegeben. Wenn jedoch keine Mount-Optionen in einer Storage-Klasse angegeben sind, stellt Astra Trident die Mount-Optionen bereit, die in der Konfigurationsdatei des Storage-Back-End angegeben sind.</p> <p>Wenn in der Storage-Klasse oder der Konfigurationsdatei keine Mount-Optionen angegeben sind, stellt Astra Trident keine Mount-Optionen für ein damit verbundener persistentes Volume fest.</p> | “ ” |
| nasType | <p>Konfiguration der Erstellung von NFS- oder SMB-Volumes</p> <p>Die Optionen lauten <code>nfs</code>, <code>smb</code>, Oder Null.</p> <p>Muss auf eingestellt sein <code>smb</code> Für SMB-Volumes. Einstellung auf null setzt standardmäßig auf NFS-Volumes.</p> | nfs |
| qtreesPerFlexvol | Maximale Ques pro FlexVol, muss im Bereich [50, 300] liegen | 200 |
| smbShare | <p>Sie können eine der folgenden Optionen angeben: Den Namen einer SMB-Freigabe, die mit der Microsoft Management Console oder der ONTAP-CLI erstellt wurde, oder einen Namen, mit dem Astra Trident die SMB-Freigabe erstellen kann.</p> <p>Dieser Parameter ist für Amazon FSX for ONTAP Back-Ends erforderlich.</p> | smb-share |

| Parameter | Beschreibung | Beispiel |
|-----------|---|--------------------|
| useREST | <p>Boolescher Parameter zur Verwendung von ONTAP REST-APIs. Technische Vorschau</p> <p>useREST Wird als Tech-Vorschau bereitgestellt, das für Testumgebungen und nicht für Produktions-Workloads empfohlen wird. Wenn eingestellt auf <code>true</code>, Astra Trident wird ONTAP REST APIs zur Kommunikation mit dem Backend verwenden.</p> <p>Diese Funktion erfordert ONTAP 9.11.1 und höher. Darüber hinaus muss die verwendete ONTAP-Login-Rolle Zugriff auf den haben <code>ontap</code> Applikation. Dies wird durch die vordefinierte zufrieden <code>vsadmin</code> Und <code>cluster-admin</code> Rollen:</p> | <code>false</code> |

Aktualisierung dataLIF Nach der Erstkonfiguration

Sie können die Daten-LIF nach der Erstkonfiguration ändern, indem Sie den folgenden Befehl ausführen, um die neue Backend-JSON-Datei mit aktualisierten Daten-LIF bereitzustellen.

```
tridentctl update backend <backend-name> -f <path-to-backend-json-file-with-updated-dataLIF>
```



Wenn PVCs an einen oder mehrere Pods angeschlossen sind, müssen Sie alle entsprechenden Pods herunterfahren und sie dann wieder zurückbringen, damit die neue logische Daten wirksam werden.

Back-End-Konfigurationsoptionen für die Bereitstellung von Volumes

Sie können die Standardbereitstellung mit diesen Optionen im `steuern defaults` Abschnitt der Konfiguration. Ein Beispiel finden Sie unten in den Konfigurationsbeispielen.

| Parameter | Beschreibung | Standard |
|------------------------------|---|-------------------|
| <code>spaceAllocation</code> | Speicherplatzzuweisung für LUNs | <code>true</code> |
| <code>spaceReserve</code> | Space Reservation Mode; „none“ (Thin) oder „Volume“ (Thick) | <code>none</code> |
| <code>snapshotPolicy</code> | Die Snapshot-Richtlinie zu verwenden | <code>none</code> |

| Parameter | Beschreibung | Standard |
|-------------------|---|--------------------------------------|
| qosPolicy | <p>QoS-Richtliniengruppe zur Zuweisung für erstellte Volumes Wählen Sie eine der qosPolicy oder adaptiveQosPolicy pro Storage-Pool oder Backend.</p> <p>Die Verwendung von QoS Policy Groups mit Astra Trident erfordert ONTAP 9.8 oder höher.</p> <p>Wir empfehlen die Verwendung einer nicht gemeinsam genutzten QoS-Richtliniengruppe und stellen sicher, dass die Richtliniengruppe auf jede Komponente einzeln angewendet wird. Eine Richtliniengruppe für Shared QoS führt zur Durchsetzung der Obergrenze für den Gesamtdurchsatz aller Workloads.</p> | “ ” |
| adaptiveQosPolicy | <p>Adaptive QoS-Richtliniengruppe mit Zuordnung für erstellte Volumes Wählen Sie eine der qosPolicy oder adaptiveQosPolicy pro Storage-Pool oder Backend.</p> <p>Nicht unterstützt durch ontap-nas-Ökonomie</p> | “ ” |
| snapshotReserve | Prozentsatz des für Snapshots reservierten Volumes „0“ | Wenn snapshotPolicy ist none, else „ |
| splitOnClone | Teilen Sie einen Klon bei der Erstellung von seinem übergeordneten Objekt auf | false |
| encryption | <p>Aktivieren Sie NetApp Volume Encryption (NVE) auf dem neuen Volume, standardmäßig aktiviert false. NVE muss im Cluster lizenziert und aktiviert sein, damit diese Option verwendet werden kann.</p> <p>Wenn NAE auf dem Backend aktiviert ist, wird jedes im Astra Trident bereitgestellte Volume NAE aktiviert.</p> <p>Weitere Informationen finden Sie unter: "Astra Trident arbeitet mit NVE und NAE zusammen".</p> | false |

| Parameter | Beschreibung | Standard |
|-----------------|--|--|
| luksEncryption | Aktivieren Sie die LUKS-Verschlüsselung. Siehe " Linux Unified Key Setup (LUKS) verwenden ". Nur SAN | " |
| tieringPolicy | Tiering-Richtlinie für die Nutzung none | snapshot-only Für Konfiguration vor ONTAP 9.5 SVM-DR |
| unixPermissions | Modus für neue Volumes. Leere leer für SMB Volumen. | " |
| securityStyle | Sicherheitstyp für neue Volumes. NFS unterstützt mixed Und unix Sicherheitsstile. SMB-Support mixed Und ntfs Sicherheitsstile. | NFS-Standard ist unix. Der SMB-Standardwert ist ntfs. |

Beispiel

Wird verwendet `nasType`, `node-stage-secret-name`, und `node-stage-secret-namespace`, Sie können ein SMB-Volume angeben und die erforderlichen Active Directory-Anmeldeinformationen angeben. SMB Volumes werden mit unterstützt `ontap-nas` Nur Treiber.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: nas-smb-sc
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-nas"
  trident.netapp.io/nasType: "smb"
  csi.storage.k8s.io/node-stage-secret-name: "smbcreds"
  csi.storage.k8s.io/node-stage-secret-namespace: "default"
```

Back-Ends mit kubectl erstellen

Ein Backend definiert die Beziehung zwischen Astra Trident und einem Storage-System. Er erzählt Astra Trident, wie man mit diesem Storage-System kommuniziert und wie Astra Trident Volumes darauf bereitstellen sollte. Nach der Installation von Astra Trident ist der nächste Schritt die Erstellung eines Backends. Der `TridentBackendConfig` Mit Custom Resource Definition (CRD) können Sie Trident Back-Ends direkt über die Kubernetes Schnittstelle erstellen und managen. Dies können Sie mit `run kubectl` Oder das vergleichbare CLI Tool für Ihre Kubernetes Distribution.

TridentBackendConfig

`TridentBackendConfig` (`tbc`, `tbconfig`, `tbackendconfig`) ist ein Front-End, Namensvetter CRD, mit dem Sie Astra Trident Back-Ends mit verwalten können `kubectl`. Kubernetes- und Storage-Administratoren können Back-Ends jetzt direkt über die Kubernetes-CLI erstellen und managen, ohne dass ein dediziertes Dienstprogramm für die Befehlszeilenschnittstelle erforderlich ist (`tridentctl`).

Bei der Erstellung eines `TridentBackendConfig` Objekt, geschieht Folgendes:

- Ein Back-End wird automatisch von Astra Trident auf Basis der von Ihnen zu erstellenden Konfiguration erstellt. Dies wird intern als A dargestellt `TridentBackend` (`tbe`, `tridentbackend`) CR.
- Der `TridentBackendConfig` ist eindeutig an A gebunden `TridentBackend` Das wurde von Astra Trident entwickelt.

Beide `TridentBackendConfig` Pflegt eine 1:1-Zuordnung mit einem `TridentBackend`. Die erstere Schnittstelle, die dem Benutzer zum Design und zur Konfiguration von Back-Ends zur Verfügung gestellt wird. Letztere ist, wie Trident das tatsächliche Backend-Objekt darstellt.



`TridentBackend` CRS werden automatisch von Astra Trident erstellt. Sie sollten diese nicht ändern. Wenn Sie an Back-Ends Aktualisierungen vornehmen möchten, ändern Sie das `TridentBackendConfig` Objekt:

Im folgenden Beispiel finden Sie Informationen zum Format des `TridentBackendConfig` CR:

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-san
spec:
  version: 1
  backendName: ontap-san-backend
  storageDriverName: ontap-san
  managementLIF: 10.0.0.1
  dataLIF: 10.0.0.2
  svm: trident_svm
  credentials:
    name: backend-tbc-ontap-san-secret
```

Sie können sich auch die Beispiele im ansehen "[trident-Installationsprogramm](#)" Verzeichnis für Beispielkonfigurationen für die gewünschte Speicherplattform/den gewünschten Service.

Der `spec` Nimmt Back-End-spezifische Konfigurationsparameter ein. In diesem Beispiel verwendet das Backend `ontap-san` Speichertreiber und verwendet die hier tabellarischen Konfigurationsparameter. Eine Liste der Konfigurationsoptionen für den gewünschten Speichertreiber finden Sie unter "[Back-End-Konfigurationsinformationen für Ihren Speichertreiber](#)".

Der `spec` Abschnitt enthält auch `credentials` Und `deletionPolicy` Felder, die neu in den eingeführt werden `TridentBackendConfig` CR:

- `credentials`: Dieser Parameter ist ein Pflichtfeld und enthält die Anmeldeinformationen, die zur Authentifizierung mit dem Speichersystem/Service verwendet werden. Dies ist auf ein vom Benutzer erstelltes Kubernetes Secret festgelegt. Die Anmeldeinformationen können nicht im Klartext weitergegeben werden und führen zu einem Fehler.
- `deletionPolicy`: Dieses Feld definiert, was passieren soll, wenn der `TridentBackendConfig` Wird gelöscht. Es kann einen von zwei möglichen Werten annehmen:
 - `delete`: Dies führt zur Löschung beider `TridentBackendConfig` CR und das zugehörige Backend. Dies ist der Standardwert.
 - `retain`: Wenn a `TridentBackendConfig` CR wird gelöscht, die Backend-Definition ist weiterhin vorhanden und kann mit verwaltet werden `tridentctl`. Einstellen der Löschrictlinie auf `retain` Benutzer können ein Downgrade auf eine frühere Version (vor 21.04) durchführen und die erstellten Back-Ends behalten. Der Wert für dieses Feld kann nach einem aktualisiert werden `TridentBackendConfig` Wird erstellt.



Der Name eines Backend wird mit festgelegt `spec.backendName`. Wenn nicht angegeben, wird der Name des Backend auf den Namen des gesetzt `TridentBackendConfig` Objekt (`metadata.name`). Es wird empfohlen, mit explizit Back-End-Namen festzulegen `spec.backendName`.



Back-Ends, die mit erstellt wurden `tridentctl` Ist nicht zugeordnet `TridentBackendConfig` Objekt: Sie können solche Back-Ends mit verwalten `kubectl` Durch Erstellen von A `TridentBackendConfig` CR. Es muss sorgfältig darauf achten, identische Konfigurationsparameter festzulegen (z. B. `spec.backendName`, `spec.storagePrefix`, `spec.storageDriverName`, Und so weiter). Astra Trident bindet automatisch die neu erstellte `TridentBackendConfig` Mit dem bereits vorhandenen Backend.

Schritte im Überblick

Um ein neues Backend mit zu erstellen `kubectl`, Sie sollten Folgendes tun:

1. Erstellen Sie ein "**Kubernetes Secret**". Das Geheimnis enthält die Zugangsdaten, die Astra Trident zur Kommunikation mit dem Storage-Cluster/Service benötigt.
2. Erstellen Sie ein `TridentBackendConfig` Objekt: Dies enthält Angaben zum Storage-Cluster/Service und verweist auf das im vorherigen Schritt erstellte Geheimnis.

Nachdem Sie ein Backend erstellt haben, können Sie den Status mit beobachten `kubectl get tbc <tbc-name> -n <trident-namespace>` Und sammeln Sie weitere Details.

Schritt: Ein Kubernetes Secret erstellen

Erstellen Sie einen geheimen Schlüssel, der die Anmeldedaten für den Zugriff für das Backend enthält. Dies ist nur bei jedem Storage Service/jeder Plattform möglich. Hier ein Beispiel:

```
kubectl -n trident create -f backend-tbc-ontap-san-secret.yaml
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-ontap-san-secret
type: Opaque
stringData:
  username: cluster-admin
  password: t@Ax@7q(>
```

In dieser Tabelle sind die Felder zusammengefasst, die für jede Speicherplattform im Secret enthalten sein müssen:

| Beschreibung der geheimen Felder der Speicherplattform | Geheim | Feldbeschreibung |
|--|----------------------|---|
| Azure NetApp Dateien | Client-ID | Die Client-ID aus einer App-Registrierung |
| Cloud Volumes Service für GCP | Private_Schlüssel_id | ID des privaten Schlüssels. Teil des API-Schlüssels für GCP-Servicekonto mit CVS-Administratorrolle |
| Cloud Volumes Service für GCP | Privater_Schlüssel | Privater Schlüssel. Teil des API-Schlüssels für GCP-Servicekonto mit CVS-Administratorrolle |
| Element (NetApp HCI/SolidFire) | Endpoint | MVIP für den SolidFire-Cluster mit Mandanten-Anmeldedaten |
| ONTAP | Benutzername | Benutzername für die Verbindung mit dem Cluster/SVM. Wird für die Anmeldeinformationsbasierte Authentifizierung verwendet |
| ONTAP | Passwort | Passwort für die Verbindung mit dem Cluster/SVM Wird für die Anmeldeinformationsbasierte Authentifizierung verwendet |
| ONTAP | KundenPrivateKey | Base64-kodierte Wert des privaten Client-Schlüssels. Wird für die zertifikatbasierte Authentifizierung verwendet |

| Beschreibung der geheimen Felder der Speicherplattform | Geheim | Feldbeschreibung |
|--|---------------------------|--|
| ONTAP | ChapUsername | Eingehender Benutzername. Erforderlich, wenn usCHAP=true verwendet wird. Für ontap-san Und ontap-san-economy |
| ONTAP | ChapInitiatorSecret | CHAP-Initiatorschlüssel. Erforderlich, wenn usCHAP=true verwendet wird. Für ontap-san Und ontap-san-economy |
| ONTAP | ChapTargetBenutzername | Zielbenutzername. Erforderlich, wenn usCHAP=true verwendet wird. Für ontap-san Und ontap-san-economy |
| ONTAP | ChapTargetInitiatorSecret | Schlüssel für CHAP-Zielinitiator. Erforderlich, wenn usCHAP=true verwendet wird. Für ontap-san Und ontap-san-economy |

Auf das in diesem Schritt erstellte Geheimnis wird im verwiesenen `spec.credentials` Feld von `TridentBackendConfig` Objekt, das im nächsten Schritt erstellt wird.

Schritt 2: Erstellen Sie die `TridentBackendConfig` CR

Sie sind jetzt bereit, Ihre zu erstellen `TridentBackendConfig` CR. In diesem Beispiel wird ein Backend verwendet, das den verwendet `ontap-san` Treiber wird mithilfe des erstellt `TridentBackendConfig` Unten gezeigte Objekte:

```
kubectl -n trident create -f backend-tbc-ontap-san.yaml
```

```

apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-san
spec:
  version: 1
  backendName: ontap-san-backend
  storageDriverName: ontap-san
  managementLIF: 10.0.0.1
  dataLIF: 10.0.0.2
  svm: trident_svm
  credentials:
    name: backend-tbc-ontap-san-secret

```

Schritt 3: Überprüfen Sie den Status des TridentBackendConfig CR

Nun, da Sie die erstellt haben TridentBackendConfig CR, Sie können den Status überprüfen. Das folgende Beispiel zeigt:

```

kubectl -n trident get tbc backend-tbc-ontap-san

```

| NAME | PHASE | STATUS | BACKEND NAME | BACKEND UUID |
|-----------------------|--------------|---------|-------------------|--------------------------|
| backend-tbc-ontap-san | bab2699e6ab8 | Bound | ontap-san-backend | 8d24fce7-6f60-4d4a-8ef6- |
| | | Success | | |

Ein Back-End wurde erfolgreich erstellt und an das gebunden TridentBackendConfig CR.

Die Phase kann einen der folgenden Werte annehmen:

- **Bound:** Das TridentBackendConfig CR ist mit einem Backend verknüpft, und dieses Backend enthält configRef Auf einstellen TridentBackendConfig CR-UID.
- **Unbound:** Dargestellt mit "". Der TridentBackendConfig Objekt ist nicht an ein Backend gebunden. Neu erstellt TridentBackendConfig CRS befinden sich standardmäßig in dieser Phase. Wenn die Phase sich ändert, kann sie nicht wieder auf Unbound zurückgesetzt werden.
- **Deleting:** Das TridentBackendConfig CR deletionPolicy Wurde auf Löschen festgelegt. Wenn der TridentBackendConfig CR wird gelöscht und wechselt in den Löschzustand.
 - Wenn im Backend keine PVCs (Persistent Volume Claims) vorhanden sind, löschen Sie den TridentBackendConfig Wird dazu führen, dass Astra Trident das Backend sowie das löscht TridentBackendConfig CR.
 - Wenn ein oder mehrere VES im Backend vorhanden sind, wechselt es in den Löschzustand. Der TridentBackendConfig Anschließend wechselt CR in die Löschphase. Das Backend und TridentBackendConfig Werden erst gelöscht, nachdem alle PVCs gelöscht wurden.
- **Lost:** Das Backend, das mit dem verbunden ist TridentBackendConfig CR wurde versehentlich oder absichtlich gelöscht und das TridentBackendConfig CR hat noch einen Verweis auf das gelöschte

Backend. Der `TridentBackendConfig` CR kann weiterhin unabhängig vom gelöscht werden `deletionPolicy` Wert:

- Unknown: Astra Trident kann den Zustand oder die Existenz des mit dem verbundenen Backend nicht bestimmen `TridentBackendConfig` CR. Beispiel: Wenn der API-Server nicht antwortet oder wenn der `tridentbackends.trident.netapp.io` CRD fehlt. Dies kann Eingriffe erfordern.

In dieser Phase wird erfolgreich ein Backend erstellt! Es gibt mehrere Operationen, die zusätzlich gehandhabt werden können, wie z. B. ["Back-End-Updates und Löschungen am Back-End"](#).

(Optional) Schritt 4: Weitere Informationen

Sie können den folgenden Befehl ausführen, um weitere Informationen über Ihr Backend zu erhalten:

```
kubectl -n trident get tbc backend-tbc-ontap-san -o wide
```

| NAME | PHASE | STATUS | STORAGE DRIVER | BACKEND NAME | DELETION POLICY | BACKEND UUID |
|-----------------------|-------|--------|----------------|-------------------|-----------------|--------------------------------------|
| backend-tbc-ontap-san | | Bound | Success | ontap-san-backend | ontap-san | 8d24fce7-6f60-4d4a-8ef6-bab2699e6ab8 |

Zusätzlich können Sie auch einen YAML/JSON Dump von erhalten `TridentBackendConfig`.

```
kubectl -n trident get tbc backend-tbc-ontap-san -o yaml
```

```

apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  creationTimestamp: "2021-04-21T20:45:11Z"
  finalizers:
  - trident.netapp.io
  generation: 1
  name: backend-tbc-ontap-san
  namespace: trident
  resourceVersion: "947143"
  uid: 35b9d777-109f-43d5-8077-c74a4559d09c
spec:
  backendName: ontap-san-backend
  credentials:
    name: backend-tbc-ontap-san-secret
  managementLIF: 10.0.0.1
  dataLIF: 10.0.0.2
  storageDriverName: ontap-san
  svm: trident_svm
  version: 1
status:
  backendInfo:
    backendName: ontap-san-backend
    backendUUID: 8d24fce7-6f60-4d4a-8ef6-bab2699e6ab8
  deletionPolicy: delete
  lastOperationStatus: Success
  message: Backend 'ontap-san-backend' created
  phase: Bound

```

`backendInfo` Enthält `backendName` Und das `backendUUID` Des Back-End, das als Antwort auf das erstellt wurde `TridentBackendConfig` CR. Der `lastOperationStatus` Feld gibt den Status des letzten Vorgangs des an `TridentBackendConfig` CR, der vom Benutzer ausgelöst werden kann (z. B. hat der Benutzer etwas in geändert `spec`) Oder ausgelöst durch Astra Trident (z. B. während Astra Trident Neustart). Er kann entweder erfolgreich oder fehlgeschlagen sein. `phase` Stellt den Status der Beziehung zwischen dem dar `TridentBackendConfig` CR und das Backend. Im obigen Beispiel `phase` Hat den Wert `gebunden`, was bedeutet, dass der `TridentBackendConfig` CR ist mit dem Backend verknüpft.

Sie können die ausführen `kubectl -n trident describe tbc <tbc-cr-name>` Befehl, um Details zu den Ereignisprotokollen zu erhalten.



Sie können ein Back-End, das einen zugeordneten enthält, nicht aktualisieren oder löschen `TridentBackendConfig` Objekt wird verwendet `tridentctl`. Um die Schritte zu verstehen, die mit dem Wechsel zwischen verbunden sind `tridentctl` Und `TridentBackendConfig`, "[Sehen Sie hier](#)".

Back-Ends managen

Führen Sie das Back-End-Management mit kubectl durch

Erfahren Sie, wie Sie mit Backend-Management-Operationen durchführen `kubectl`.

Löschen Sie ein Back-End

Durch Löschen von `A TridentBackendConfig`, Sie weisen Astra Trident an, Back-Ends zu löschen/zu behalten (basierend auf `deletionPolicy`). Um ein Backend zu löschen, stellen Sie sicher, dass `deletionPolicy` Ist auf Löschen festgelegt. Um nur die zu löschen `TridentBackendConfig`, Stellen Sie das sicher `deletionPolicy` Auf beibehalten eingestellt. Dadurch wird sichergestellt, dass das Backend weiterhin vorhanden ist und mit verwaltet werden kann `tridentctl`.

Führen Sie den folgenden Befehl aus:

```
kubectl delete tbc <tbc-name> -n trident
```

Astra Trident löscht nicht die Kubernetes Secrets, die von verwendet wurden `TridentBackendConfig`. Der Kubernetes-Benutzer ist für die Bereinigung von Geheimnissen verantwortlich. Beim Löschen von Geheimnissen ist Vorsicht zu nehmen. Sie sollten Geheimnisse nur löschen, wenn sie nicht von den Back-Ends verwendet werden.

Zeigen Sie die vorhandenen Back-Ends an

Führen Sie den folgenden Befehl aus:

```
kubectl get tbc -n trident
```

Sie können auch ausführen `tridentctl get backend -n trident` Oder `tridentctl get backend -o yaml -n trident` Um eine Liste aller vorhandenen Back-Ends zu erhalten. Diese Liste umfasst auch Back-Ends, die mit erstellt wurden `tridentctl`.

Aktualisieren Sie ein Backend

Es gibt mehrere Gründe für die Aktualisierung eines Backend:

- Die Anmeldeinformationen für das Speichersystem wurden geändert. Um Anmeldedaten zu aktualisieren, wird das in verwendete Kubernetes Secret verwendet `TridentBackendConfig` Objekt muss aktualisiert werden. Astra Trident aktualisiert automatisch das Backend mit den neuesten Zugangsdaten. Führen Sie den folgenden Befehl aus, um den Kubernetes Secret zu aktualisieren:

```
kubectl apply -f <updated-secret-file.yaml> -n trident
```

- Parameter (wie der Name der verwendeten ONTAP-SVM) müssen aktualisiert werden.
 - Sie können aktualisieren `TridentBackendConfig` Objekte können direkt über Kubernetes mit dem folgenden Befehl abgerufen werden:

```
kubectl apply -f <updated-backend-file.yaml>
```

- Alternativ können Sie Änderungen an der vorhandenen vornehmen `TridentBackendConfig` CR mit folgendem Befehl:

```
kubectl edit tbc <tbc-name> -n trident
```



- Wenn ein Backend-Update fehlschlägt, bleibt das Backend in seiner letzten bekannten Konfiguration erhalten. Sie können die Protokolle anzeigen, um die Ursache durch Ausführen zu bestimmen `kubectl get tbc <tbc-name> -o yaml -n trident` Oder `kubectl describe tbc <tbc-name> -n trident`.
- Nachdem Sie das Problem mit der Konfigurationsdatei erkannt und behoben haben, können Sie den Befehl Update erneut ausführen.

Back-End-Management mit `tridentctl`

Erfahren Sie, wie Sie mit Backend-Management-Operationen durchführen `tridentctl`.

Erstellen Sie ein Backend

Nachdem Sie ein erstellt haben "[Back-End-Konfigurationsdatei](#)", Ausführen des folgenden Befehls:

```
tridentctl create backend -f <backend-file> -n trident
```

Wenn die Back-End-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 logs -n trident
```

Nachdem Sie das Problem mit der Konfigurationsdatei identifiziert und behoben haben, können Sie einfach die ausführen `create` Befehl erneut.

Löschen Sie ein Back-End

Gehen Sie wie folgt vor, um ein Backend von Astra Trident zu löschen:

1. Abrufen des Back-End-Namens:

```
tridentctl get backend -n trident
```

2. Back-End löschen:


```
tridentctl delete backend <backend-name> -n trident
```



Wenn Astra Trident Volumes und Snapshots aus diesem Backend bereitgestellt hat, die immer noch vorhanden sind, verhindert das Löschen des Backend, dass neue Volumes bereitgestellt werden. Das Backend wird weiterhin in einem „Deleting“ Zustand vorhanden sein und Trident wird weiterhin diese Volumes und Snapshots verwalten, bis sie gelöscht werden.

Zeigen Sie die vorhandenen Back-Ends an

Gehen Sie zum Anzeigen der von Trident verwendeten Back-Ends wie folgt vor:

- Führen Sie den folgenden Befehl aus, um eine Zusammenfassung anzuzeigen:

```
tridentctl get backend -n trident
```

- Um alle Details anzuzeigen, führen Sie den folgenden Befehl aus:

```
tridentctl get backend -o json -n trident
```

Aktualisieren Sie ein Backend

Führen Sie nach dem Erstellen einer neuen Backend-Konfigurationsdatei den folgenden Befehl aus:

```
tridentctl update backend <backend-name> -f <backend-file> -n trident
```

Wenn das Backend-Update fehlschlägt, ist bei der Backend-Konfiguration ein Fehler aufgetreten oder Sie haben ein ungültiges Update versucht. Sie können die Protokolle zur Bestimmung der Ursache anzeigen, indem Sie den folgenden Befehl ausführen:

```
tridentctl logs -n trident
```

Nachdem Sie das Problem mit der Konfigurationsdatei identifiziert und behoben haben, können Sie einfach die `update` Befehl erneut.

Identifizieren Sie die Storage-Klassen, die ein Backend nutzen

Dies ist ein Beispiel für die Art von Fragen, die Sie mit der JSON beantworten können `tridentctl` Ausgänge für Backend-Objekte. Dazu wird der verwendet `jq` Dienstprogramm, das Sie installieren müssen.

```
tridentctl get backend -o json | jq '[.items[] | {backend: .name, storageClasses: [.storage[].storageClasses]|unique}]'
```

Dies gilt auch für Back-Ends, die mit erstellt wurden `TridentBackendConfig`.

Wechseln Sie zwischen den Back-End-Managementoptionen

Erfahren Sie in Astra Trident, wie Back-Ends auf verschiedene Art und Weise gemanagt werden.

Optionen für das Management von Back-Ends

Mit der Einführung von `TridentBackendConfig`, Administratoren haben jetzt zwei unterschiedliche Arten von Back-Ends zu verwalten. Dies stellt die folgenden Fragen:

- Mit können Back-Ends erstellt werden `tridentctl` Gemanagt werden mit `TridentBackendConfig`?
- Mit können Back-Ends erstellt werden `TridentBackendConfig` Gemanagt werden mit `tridentctl`?

Managen `tridentctl` Back-Ends mit `TridentBackendConfig`

In diesem Abschnitt werden die Schritte aufgeführt, die für das Management von Back-Ends erforderlich sind, die mit erstellt wurden `tridentctl` Erstellen Sie direkt über die Kubernetes Schnittstelle `TridentBackendConfig` Objekte:

Dies gilt für die folgenden Szenarien:

- Bereits vorhandene Back-Ends, die keine haben `TridentBackendConfig` Weil sie mit erstellt wurden `tridentctl`.
- Neue Back-Ends, mit denen erstellt wurden `tridentctl`, Während andere `TridentBackendConfig` Objekte sind vorhanden.

In beiden Szenarien werden Back-Ends weiterhin vorhanden sein, wobei Astra Trident Volumes terminieren und darauf arbeiten wird. Administratoren können hier eine von zwei Möglichkeiten wählen:

- Fahren Sie mit der Verwendung fort `tridentctl` Um Back-Ends zu managen, die mit ihr erstellt wurden.
- Back-Ends werden mit erstellt `tridentctl` Zu einer neuen `TridentBackendConfig` Objekt: Dies würde bedeuten, dass die Back-Ends mit gemanagt werden `kubect1` Und nicht `tridentctl`.

Um ein bereits vorhandenes Backend mit zu verwalten `kubect1`, Sie müssen ein erstellen `TridentBackendConfig` Das bindet an das vorhandene Backend. Hier eine Übersicht über die Funktionsweise:

1. Kubernetes Secret erstellen: Das Geheimnis enthält die Zugangsdaten, die Astra Trident zur Kommunikation mit dem Storage-Cluster/Service benötigt.
2. Erstellen Sie ein `TridentBackendConfig` Objekt: Dies enthält Angaben zum Storage-Cluster/Service und verweist auf das im vorherigen Schritt erstellte Geheimnis. Es muss sorgfältig darauf achten, identische Konfigurationsparameter festzulegen (z. B. `spec.backendName`, `spec.storagePrefix`, `spec.storageDriverName`, Und so weiter). `spec.backendName` Muss auf den Namen des vorhandenen Backend eingestellt werden.

Schritt 0: Identifizieren Sie das Backend

Um ein zu erstellen `TridentBackendConfig` Die an ein vorhandenes Backend bindet, müssen Sie die Backend-Konfiguration abrufen. In diesem Beispiel nehmen wir an, dass ein Backend mithilfe der folgenden

JSON-Definition erstellt wurde:

```
tridentctl get backend ontap-nas-backend -n trident
+-----+-----+
+-----+-----+-----+-----+
|          NAME          | STORAGE DRIVER |          UUID          |
| STATE | VOLUMES |
+-----+-----+
+-----+-----+-----+-----+
| ontap-nas-backend    | ontap-nas      | 52f2eb10-e4c6-4160-99fc-
96b3be5ab5d7 | online |      25 |
+-----+-----+-----+-----+
+-----+-----+-----+-----+

cat ontap-nas-backend.json

{
  "version": 1,
  "storageDriverName": "ontap-nas",
  "managementLIF": "10.10.10.1",
  "dataLIF": "10.10.10.2",
  "backendName": "ontap-nas-backend",
  "svm": "trident_svm",
  "username": "cluster-admin",
  "password": "admin-password",

  "defaults": {
    "spaceReserve": "none",
    "encryption": "false"
  },
  "labels": {"store": "nas_store"},
  "region": "us_east_1",
  "storage": [
    {
      "labels": {"app": "msoffice", "cost": "100"},
      "zone": "us_east_1a",
      "defaults": {
        "spaceReserve": "volume",
        "encryption": "true",
        "unixPermissions": "0755"
      }
    },
    {
      "labels": {"app": "mysqldb", "cost": "25"},
      "zone": "us_east_1d",
      "defaults": {
```

```

        "spaceReserve": "volume",
        "encryption": "false",
        "unixPermissions": "0775"
    }
}
]
}

```

Schritt: Ein Kubernetes Secret erstellen

Erstellen Sie einen geheimen Schlüssel, der die Anmeldeinformationen für das Backend enthält, wie in diesem Beispiel gezeigt:

```

cat tbc-ontap-nas-backend-secret.yaml

apiVersion: v1
kind: Secret
metadata:
  name: ontap-nas-backend-secret
type: Opaque
stringData:
  username: cluster-admin
  password: admin-password

kubectl create -f tbc-ontap-nas-backend-secret.yaml -n trident
secret/backend-tbc-ontap-san-secret created

```

Schritt 2: Erstellen Sie ein TridentBackendConfig CR

Im nächsten Schritt wird ein erstellt `TridentBackendConfig` CR, das automatisch an die bereits vorhandene bindet `ontap-nas-backend` (Wie in diesem Beispiel). Stellen Sie sicher, dass folgende Anforderungen erfüllt sind:

- Der gleiche Backend-Name wird in definiert `spec.backendName`.
- Die Konfigurationsparameter sind mit dem ursprünglichen Back-End identisch.
- Virtuelle Pools (falls vorhanden) müssen dieselbe Reihenfolge wie im ursprünglichen Backend beibehalten.
- Anmeldedaten werden bei einem Kubernetes Secret und nicht im Klartext bereitgestellt.

In diesem Fall die `TridentBackendConfig` Wird so aussehen:

```

cat backend-tbc-ontap-nas.yaml
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: tbc-ontap-nas-backend
spec:
  version: 1
  storageDriverName: ontap-nas
  managementLIF: 10.10.10.1
  dataLIF: 10.10.10.2
  backendName: ontap-nas-backend
  svm: trident_svm
  credentials:
    name: mysecret
  defaults:
    spaceReserve: none
    encryption: 'false'
  labels:
    store: nas_store
  region: us_east_1
  storage:
  - labels:
    app: msoffice
    cost: '100'
    zone: us_east_1a
    defaults:
      spaceReserve: volume
      encryption: 'true'
      unixPermissions: '0755'
  - labels:
    app: mysqldb
    cost: '25'
    zone: us_east_1d
    defaults:
      spaceReserve: volume
      encryption: 'false'
      unixPermissions: '0775'

kubectl create -f backend-tbc-ontap-nas.yaml -n trident
tridentbackendconfig.trident.netapp.io/tbc-ontap-nas-backend created

```

Schritt 3: Überprüfen Sie den Status des TridentBackendConfig **CR**

Nach dem TridentBackendConfig Wurde erstellt, seine Phase muss sein Bound. Sie sollte außerdem den gleichen Backend-Namen und die gleiche UUID wie das vorhandene Backend widerspiegeln.

```
kubectl get tbc tbc-ontap-nas-backend -n trident
NAME                                BACKEND NAME                BACKEND UUID
PHASE    STATUS
tbc-ontap-nas-backend  ontap-nas-backend          52f2eb10-e4c6-4160-99fc-
96b3be5ab5d7    Bound    Success
```

#confirm that no new backends were created (i.e., TridentBackendConfig did not end up creating a new backend)

```
tridentctl get backend -n trident
```

```
+-----+-----+
+-----+-----+-----+-----+
|          NAME          | STORAGE DRIVER |          UUID          |
| STATE  | VOLUMES |          |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
| ontap-nas-backend     | ontap-nas      | 52f2eb10-e4c6-4160-99fc-
96b3be5ab5d7 | online |          25 |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
```

Das Backend wird nun vollständig mit dem verwaltet tbc-ontap-nas-backend TridentBackendConfig Objekt:

Managen TridentBackendConfig Back-Ends mit tridentctl

```
`tridentctl` Kann zur Auflistung von Back-Ends verwendet werden, die mit
erstellt wurden `TridentBackendConfig`. Darüber hinaus können
Administratoren solche Back-Ends mithilfe von auch vollständig managen
`tridentctl` Durch Löschen `TridentBackendConfig` Mit Sicherheit
`spec.deletionPolicy` Ist auf festgelegt `retain`.
```

Schritt 0: Identifizieren Sie das Backend

Nehmen wir beispielsweise an, dass das folgende Backend mit erstellt wurde TridentBackendConfig:

```
kubectl get tbc backend-tbc-ontap-san -n trident -o wide
NAME                                BACKEND NAME                BACKEND UUID
PHASE  STATUS    STORAGE DRIVER  DELETION POLICY
backend-tbc-ontap-san  ontap-san-backend  81abcb27-ea63-49bb-b606-
0a5315ac5f82    Bound    Success    ontap-san        delete

tridentctl get backend ontap-san-backend -n trident
+-----+-----+
+-----+-----+-----+-----+
|          NAME          | STORAGE DRIVER |                               UUID
| STATE  | VOLUMES |
+-----+-----+
+-----+-----+-----+-----+
| ontap-san-backend | ontap-san      | 81abcb27-ea63-49bb-b606-
0a5315ac5f82 | online |          33 |
+-----+-----+
+-----+-----+-----+-----+
```

Von der Ausgabe, ist es gesehen, dass TridentBackendConfig Wurde erfolgreich erstellt und ist an ein Backend gebunden [UUID des Backends beobachten].

Schritt 1: Bestätigen deletionPolicy ist auf festgelegt retain

Lassen Sie uns den Wert von betrachten deletionPolicy. Dies muss eingestellt werden retain. Dadurch wird sichergestellt, dass, wenn ein TridentBackendConfig CR wird gelöscht, die Backend-Definition ist weiterhin vorhanden und kann mit verwaltet werden tridentctl.

```
kubectl get tbc backend-tbc-ontap-san -n trident -o wide
NAME                                BACKEND NAME                BACKEND UUID
PHASE  STATUS    STORAGE DRIVER  DELETION POLICY
backend-tbc-ontap-san  ontap-san-backend  81abcb27-ea63-49bb-b606-
0a5315ac5f82    Bound    Success    ontap-san        delete

# Patch value of deletionPolicy to retain
kubectl patch tbc backend-tbc-ontap-san --type=merge -p
'{"spec":{"deletionPolicy":"retain"}}' -n trident
tridentbackendconfig.trident.netapp.io/backend-tbc-ontap-san patched

#Confirm the value of deletionPolicy
kubectl get tbc backend-tbc-ontap-san -n trident -o wide
NAME                                BACKEND NAME                BACKEND UUID
PHASE  STATUS    STORAGE DRIVER  DELETION POLICY
backend-tbc-ontap-san  ontap-san-backend  81abcb27-ea63-49bb-b606-
0a5315ac5f82    Bound    Success    ontap-san        retain
```



Fahren Sie nur mit dem nächsten Schritt fort `deletionPolicy` ist auf festgelegt `retain`.

Schritt 2: Löschen Sie den `TridentBackendConfig` CR

Der letzte Schritt besteht darin, den zu löschen `TridentBackendConfig` CR. Nach Bestätigung des `deletionPolicy` ist auf festgelegt `retain`, Sie können mit der Löschung fortfahren:

```
kubectl delete tbc backend-tbc-ontap-san -n trident
tridentbackendconfig.trident.netapp.io "backend-tbc-ontap-san" deleted

tridentctl get backend ontap-san-backend -n trident
+-----+-----+-----+
+-----+-----+-----+
|          NAME          | STORAGE DRIVER |          UUID          |
| STATE | VOLUMES |          |
+-----+-----+-----+
+-----+-----+-----+
| ontap-san-backend | ontap-san      | 81abcb27-ea63-49bb-b606-
0a5315ac5f82 | online |          33 |
+-----+-----+-----+
+-----+-----+-----+
```

Bei der Löschung der `TridentBackendConfig` Object, Astra Trident entfernt es einfach, ohne das Backend zu löschen.

Erstellen und Managen von Storage-Klassen

Erstellen Sie eine Speicherklasse

Konfigurieren Sie ein Kubernetes `StorageClass`-Objekt und erstellen Sie die Storage-Klasse, um Astra Trident über die Bereitstellung von Volumes zu informieren.

Konfigurieren Sie ein Kubernetes `StorageClass`-Objekt

Der "[Kubernetes StorageClass-Objekt](#)" Astra Trident wird als bereitstellung identifiziert, die für diese Klasse verwendet wird und Astra Trident zur Bereitstellung eines Volumes anweist. Beispiel:


```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: <Name>
provisioner: csi.trident.netapp.io
mountOptions: <Mount Options>
parameters:
  <Trident Parameters>
allowVolumeExpansion: true
volumeBindingMode: Immediate
```

Siehe "[Kubernetes und Trident Objekte](#)" Erfahren Sie, wie Storage-Klassen mit dem interagieren PersistentVolumeClaim Und Parameter für die Steuerung, wie Astra Trident Volumes provisioniert.

Erstellen Sie eine Speicherklasse

Nachdem Sie das StorageClass-Objekt erstellt haben, können Sie die Storage-Klasse erstellen. [Proben der Lagerklasse](#) Enthält einige grundlegende Proben, die Sie verwenden oder ändern können.

Schritte

1. Verwenden Sie dieses Objekt von Kubernetes `kubectl` Um sie in Kubernetes zu erstellen.

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

2. 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"
        ]
      }
    }
  ]
}

```

Proben der Lagerklasse

Astra Trident bietet das ["Einfache Definitionen von Storage-Klassen für spezifische Back-Ends"](#).

Alternativ können Sie bearbeiten `sample-input/storage-class-csi.yaml.templ` Datei, die im Lieferumfang des Installationsprogramms enthalten ist und ersetzt wird `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

```

Management von Storage-Klassen

Sie können vorhandene Storage-Klassen anzeigen, eine Standard-Storage-Klasse festlegen, das Back-End der Speicherklasse identifizieren und Speicherklassen löschen.

Sehen Sie sich die vorhandenen Speicherklassen an

- Um vorhandene Kubernetes-Storage-Klassen anzuzeigen, führen Sie den folgenden Befehl aus:

```
kubectl get storageclass
```

- Um die Details der Kubernetes-Storage-Klasse anzuzeigen, führen Sie den folgenden Befehl aus:

```
kubectl get storageclass <storage-class> -o json
```

- Führen Sie den folgenden Befehl aus, um die synchronisierten Storage-Klassen von Astra Trident anzuzeigen:

```
tridentctl get storageclass
```

- Um die synchronisierten Storage-Klassendetails von Astra Trident anzuzeigen, führen Sie den folgenden Befehl aus:

```
tridentctl get storageclass <storage-class> -o json
```

Legen Sie eine Standard-speicherklasse fest

Mit Kubernetes 1.6 können Sie eine Standard-Storage-Klasse festlegen. Dies ist die Storage-Klasse, die zur Bereitstellung eines Persistent Volume verwendet wird, wenn ein Benutzer in einer Persistent Volume Claim (PVC) nicht eine Angabe vorgibt.

- Definieren Sie eine Standard-Storage-Klasse, indem Sie die Anmerkung festlegen `storageclass.kubernetes.io/is-default-class` In der Definition der Storage-Klassen wie den „true“. Gemäß der Spezifikation wird jeder andere Wert oder jede Abwesenheit der Anmerkung als falsch interpretiert.
- Sie können eine vorhandene Storage-Klasse als Standard-Storage-Klasse konfigurieren, indem Sie den folgenden Befehl verwenden:

```
kubectl patch storageclass <storage-class-name> -p '{"metadata": {"annotations":{"storageclass.kubernetes.io/is-default-class":"true"}}}'
```

- In ähnlicher Weise können Sie die standardmäßige Storage-Klassenbeschriftung mithilfe des folgenden Befehls entfernen:

```
kubectl patch storageclass <storage-class-name> -p '{"metadata": {"annotations":{"storageclass.kubernetes.io/is-default-class":"false"}}}'
```

Es gibt auch Beispiele im Trident Installationspaket, die diese Annotation enthält.



Ihr Cluster sollte immer nur eine Standard-Storage-Klasse aufweisen. Kubernetes verhindert technisch nicht, dass Sie mehr als eine haben, aber es verhält sich so, als ob es überhaupt keine Standard-Storage-Klasse gibt.

Das Backend für eine Storage-Klasse ermitteln

Dies ist ein Beispiel für die Art von Fragen, die Sie mit der JSON beantworten können `tridentctl` Ausgänge für Astra Trident Backend-Objekte. Dazu wird der verwendet `jq` Dienstprogramm, das Sie möglicherweise zuerst installieren müssen.

```
tridentctl get storageclass -o json | jq '[.items[] | {storageClass: .Config.name, backends: [.storage]|unique}]'
```

Löschen Sie eine Speicherklasse

Führen Sie den folgenden Befehl aus, um eine Storage-Klasse aus Kubernetes zu löschen:

```
kubectl delete storageclass <storage-class>
```

<storage-class> Sollten durch Ihre Storage-Klasse ersetzt werden.

Alle persistenten Volumes, die durch diese Storage-Klasse erstellt wurden, werden unverändert beibehalten und Astra Trident wird sie weiterhin managen.



Astra Trident setzt ein Leereinschub um `fsType` Für die von ihm erstellten Volumes. Bei iSCSI-Back-Ends wird die Durchsetzung empfohlen `parameters.fsType` In der `StorageClass`. Sie sollten vorhandene `StorageClasses` löschen und mit neu erstellen `parameters.fsType` Angegeben.

Provisionierung und Management von Volumes

Bereitstellen eines Volumes

Erstellen Sie ein `PersistentVolume` (PV) und ein `PersistentVolumeClaim` (PVC), das die konfigurierte Kubernetes `StorageClass` verwendet, um Zugriff auf das PV anzufordern. Anschließend können Sie das PV an einem Pod montieren.

Überblick

A "*PersistentVolume*" (PV) ist eine physische Speicherressource, die vom Clusteradministrator auf einem Kubernetes-Cluster bereitgestellt wird. Der "*PersistentVolumeClaim*" (PVC) ist eine Anforderung für den Zugriff auf das `PersistentVolume` auf dem Cluster.

Die PVC kann so konfiguriert werden, dass eine Speicherung einer bestimmten Größe oder eines bestimmten Zugriffsmodus angefordert wird. Mithilfe der zugehörigen `StorageClass` kann der Clusteradministrator mehr als die Größe des `PersistentVolume` und den Zugriffsmodus steuern, z. B. die Performance oder das Service-Level.

Nachdem Sie das PV und die PVC erstellt haben, können Sie das Volume in einem Pod einbinden.

Beispielmanifeste

PersistentVolume-Beispielmanifest

Dieses Beispielmanifest zeigt ein Basis-PV von 10Gi, das mit StorageClass verknüpft ist `basic-csi`.

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-storage
  labels:
    type: local
spec:
  storageClassName: basic-csi
  capacity:
    storage: 10Gi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: "/my/host/path"
```

PersistentVolumeClaim-Beispielmanifest

Dieses Beispiel zeigt eine grundlegende PVC mit RWO-Zugriff, die einer StorageClass mit dem Namen zugeordnet ist `basic-csi`.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc-storage
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: basic-csi
```

Pod-Manifest-Beispiel

```
kind: Pod
apiVersion: v1
metadata:
  name: pv-pod
spec:
  volumes:
    - name: pv-storage
      persistentVolumeClaim:
        claimName: basic
  containers:
    - name: pv-container
      image: nginx
      ports:
        - containerPort: 80
          name: "http-server"
      volumeMounts:
        - mountPath: "/my/mount/path"
          name: pv-storage
```

Erstellen Sie das PV und die PVC

Schritte

1. Erstellen Sie das PV.

```
kubectl create -f pv.yaml
```

2. Überprüfen Sie den PV-Status.

```
kubectl get pv
NAME          CAPACITY  ACCESS MODES  RECLAIM POLICY  STATUS  CLAIM
STORAGECLASS  REASON    AGE
pv-storage    4Gi       RWO           Retain          Available
7s
```

3. Erstellen Sie die PVC.

```
kubectl create -f pvc.yaml
```

4. Überprüfen Sie den PVC-Status.

```
kubectl get pvc
NAME          STATUS VOLUME          CAPACITY ACCESS MODES STORAGECLASS AGE
pvc-storage  Bound  pv-name  2Gi          RWO                                     5m
```

5. Mounten Sie das Volume in einem Pod.

```
kubectl create -f pv-pod.yaml
```



Sie können den Fortschritt mit überwachen `kubectl get pod --watch`.

6. Vergewissern Sie sich, dass das Volume auf gemountet ist `/my/mount/path`.

```
kubectl exec -it task-pv-pod -- df -h /my/mount/path
```

7. Sie können den Pod jetzt löschen. Die Pod Applikation wird nicht mehr existieren, aber das Volume bleibt erhalten.

```
kubectl delete pod task-pv-pod
```

Siehe "[Kubernetes und Trident Objekte](#)" Erfahren Sie, wie Storage-Klassen mit dem interagieren PersistentVolumeClaim Und Parameter für die Steuerung, wie Astra Trident Volumes provisioniert.

Erweitern Sie Volumes

Astra Trident bietet Kubernetes-Benutzern die Möglichkeit, ihre Volumes nach Erstellung zu erweitern. Hier finden Sie Informationen zu den erforderlichen Konfigurationen zum erweitern von iSCSI- und NFS-Volumes.

Erweitern Sie ein iSCSI-Volume

Sie können ein iSCSI Persistent Volume (PV) mithilfe der CSI-provisionierung erweitern.



Die Erweiterung des iSCSI-Volumes wird von unterstützt `ontap-san`, `ontap-san-economy`, `solidfire-san` Treiber und erfordert Kubernetes 1.16 und höher.

Schritt: Storage Class für Volume-Erweiterung konfigurieren

Bearbeiten Sie die StorageClass-Definition, um die festzulegen `allowVolumeExpansion` Feld an `true`.


```
cat storageclass-ontapsan.yaml
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-san
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
allowVolumeExpansion: True
```

Bearbeiten Sie für eine bereits vorhandene StorageClass, um die einzuschließen `allowVolumeExpansion` Parameter.

Schritt 2: Erstellen Sie ein PVC mit der von Ihnen erstellten StorageClass

Bearbeiten Sie die PVC-Definition, und aktualisieren Sie die `spec.resources.requests.storage` Um die neu gewünschte Größe zu reflektieren, die größer als die ursprüngliche Größe sein muss.

```
cat pvc-ontapsan.yaml
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: san-pvc
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: ontap-san
```

Astra Trident erstellt ein persistentes Volume (PV) und verknüpft es mit dieser Persistent Volume Claim (PVC).

```

kubect1 get pvc
NAME          STATUS    VOLUME                                     CAPACITY
ACCESS MODES  STORAGECLASS  AGE
san-pvc      Bound      pvc-8a814d62-bd58-4253-b0d1-82f2885db671  1Gi
RWO          ontap-san    8s

kubect1 get pv
NAME          CAPACITY  ACCESS MODES
RECLAIM POLICY  STATUS    CLAIM          STORAGECLASS  REASON  AGE
pvc-8a814d62-bd58-4253-b0d1-82f2885db671  1Gi      RWO
Delete        Bound      default/san-pvc  ontap-san    10s

```

Schritt 3: Definieren Sie einen Behälter, der das PVC befestigt

Schließen Sie das PV an einen Pod an, um die Größe zu ändern. Beim Ändern der Größe eines iSCSI-PV gibt es zwei Szenarien:

- Wenn das PV an einen POD angeschlossen ist, erweitert Astra Trident das Volume auf dem Storage-Backend, setzt das Gerät neu ein und vergrößert das Dateisystem neu.
- Bei dem Versuch, die Größe eines nicht angeschlossenen PV zu ändern, erweitert Astra Trident das Volume auf dem Storage-Backend. Nachdem die PVC an einen Pod gebunden ist, lässt Trident das Gerät neu in die Größe des Dateisystems einarbeiten. Kubernetes aktualisiert dann die PVC-Größe, nachdem der Expand-Vorgang erfolgreich abgeschlossen ist.

In diesem Beispiel wird ein POD erstellt, der die verwendet `san-pvc`.

```
kubectl get pod
NAME          READY   STATUS    RESTARTS   AGE
ubuntu-pod   1/1     Running   0           65s

kubectl describe pvc san-pvc
Name:          san-pvc
Namespace:     default
StorageClass:  ontap-san
Status:        Bound
Volume:        pvc-8a814d62-bd58-4253-b0d1-82f2885db671
Labels:        <none>
Annotations:   pv.kubernetes.io/bind-completed: yes
               pv.kubernetes.io/bound-by-controller: yes
               volume.beta.kubernetes.io/storage-provisioner:
               csi.trident.netapp.io
Finalizers:    [kubernetes.io/pvc-protection]
Capacity:      1Gi
Access Modes:  RWO
VolumeMode:    Filesystem
Mounted By:    ubuntu-pod
```

Schritt 4: Erweitern Sie das PV

Um die Größe des PV zu ändern, das von 1Gi auf 2Gi erstellt wurde, bearbeiten Sie die PVC-Definition und aktualisieren Sie die `spec.resources.requests.storage` Bis 2Gi.

```
kubectl edit pvc san-pvc
# Please edit the object below. Lines beginning with a '#' will be
ignored,
# and an empty file will abort the edit. If an error occurs while saving
this file will be
# reopened with the relevant failures.
#
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    pv.kubernetes.io/bind-completed: "yes"
    pv.kubernetes.io/bound-by-controller: "yes"
    volume.beta.kubernetes.io/storage-provisioner: csi.trident.netapp.io
  creationTimestamp: "2019-10-10T17:32:29Z"
  finalizers:
  - kubernetes.io/pvc-protection
  name: san-pvc
  namespace: default
  resourceVersion: "16609"
  selfLink: /api/v1/namespaces/default/persistentvolumeclaims/san-pvc
  uid: 8a814d62-bd58-4253-b0d1-82f2885db671
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 2Gi
  ...
```

Schritt 5: Validieren Sie die Erweiterung

Sie können die korrekte Ausführung der Erweiterung überprüfen, indem Sie die Größe der PVC, PV und des Astra Trident Volume überprüfen:

```

kubect1 get pvc san-pvc
NAME          STATUS    VOLUME                                     CAPACITY
ACCESS MODES  STORAGECLASS  AGE
san-pvc      Bound      pvc-8a814d62-bd58-4253-b0d1-82f2885db671  2Gi
RWO          ontap-san    11m
kubect1 get pv
NAME          CAPACITY  ACCESS MODES
RECLAIM POLICY  STATUS    CLAIM          STORAGECLASS  REASON  AGE
pvc-8a814d62-bd58-4253-b0d1-82f2885db671  2Gi      RWO
Delete          Bound      default/san-pvc  ontap-san    12m
tridentctl get volumes -n trident
+-----+-----+-----+
+-----+-----+-----+-----+
|          NAME          |  SIZE  | STORAGE CLASS |
PROTOCOL |          BACKEND UUID          |  STATE  | MANAGED |
+-----+-----+-----+
+-----+-----+-----+-----+
| pvc-8a814d62-bd58-4253-b0d1-82f2885db671 | 2.0 GiB | ontap-san    |
block    | a9b7bfff-0505-4e31-b6c5-59f492e02d33 | online | true    |
+-----+-----+-----+
+-----+-----+-----+-----+

```

Erweitern Sie ein NFS-Volumen

Astra Trident unterstützt die Volume-Erweiterung für auf bereitgestellte NFS PVS `ontap-nas`, `ontap-nas-economy`, `ontap-nas-flexgroup`, `gcp-cvs`, und `azure-netapp-files` Back-Ends:

Schritt: Storage Class für Volume-Erweiterung konfigurieren

Um die Größe eines NFS PV zu ändern, muss der Administrator zunächst die Storage-Klasse konfigurieren, um die Volume-Erweiterung durch Einstellen der zu ermöglichen `allowVolumeExpansion` Feld an `true`:

```

cat storageclass-ontapnas.yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontapnas
provisioner: csi.trident.netapp.io
parameters:
  backendType: ontap-nas
allowVolumeExpansion: true

```

Wenn Sie bereits eine Storage-Klasse ohne diese Option erstellt haben, können Sie die vorhandene Storage-Klasse einfach mit `kubect1 edit storageclass` bearbeiten, um eine Volume-Erweiterung zu ermöglichen.

Schritt 2: Erstellen Sie ein PVC mit der von Ihnen erstellten StorageClass

```
cat pvc-ontapnas.yaml
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: ontapnas20mb
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 20Mi
  storageClassName: ontapnas
```

Astra Trident sollte ein 20MiB NFS PV für diese PVC erstellen:

```
kubectl get pvc
NAME                STATUS    VOLUME
CAPACITY            ACCESS MODES  STORAGECLASS  AGE
ontapnas20mb       Bound     pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7  20Mi
RWO                 ontapnas          9s

kubectl get pv pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7
NAME                CAPACITY  ACCESS MODES
RECLAIM POLICY     STATUS    CLAIM                STORAGECLASS  REASON
AGE
pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7  20Mi     RWO
Delete            Bound     default/ontapnas20mb  ontapnas
2m42s
```

Schritt 3: Erweitern Sie das PV

Um die Größe des neu erstellten 20MiB PV auf 1 gib zu ändern, bearbeiten Sie die PVC und den Satz `spec.resources.requests.storage` Bis 1 GB:

```

kubect1 edit pvc ontapnas20mb
# Please edit the object below. Lines beginning with a '#' will be
ignored,
# and an empty file will abort the edit. If an error occurs while saving
this file will be
# reopened with the relevant failures.
#
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    pv.kubernetes.io/bind-completed: "yes"
    pv.kubernetes.io/bound-by-controller: "yes"
    volume.beta.kubernetes.io/storage-provisioner: csi.trident.netapp.io
  creationTimestamp: 2018-08-21T18:26:44Z
  finalizers:
  - kubernetes.io/pvc-protection
  name: ontapnas20mb
  namespace: default
  resourceVersion: "1958015"
  selfLink: /api/v1/namespaces/default/persistentvolumeclaims/ontapnas20mb
  uid: c1bd7fa5-a56f-11e8-b8d7-fa163e59eaab
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  ...

```

Schritt 4: Validierung der Erweiterung

Sie können die korrekte Größenänderung validieren, indem Sie die Größe des PVC, des PV und des Astra Trident Volume überprüfen:

```

kubect1 get pvc ontapnas20mb
NAME                STATUS      VOLUME
CAPACITY    ACCESS MODES   STORAGECLASS   AGE
ontapnas20mb    Bound        pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7    1Gi
RWO                ontapnas                4m44s

kubect1 get pv pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7
NAME                CAPACITY    ACCESS MODES
RECLAIM POLICY     STATUS      CLAIM                STORAGECLASS   REASON
AGE
pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7    1Gi                RWO
Delete                Bound        default/ontapnas20mb    ontapnas
5m35s

tridentctl get volume pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7 -n trident
+-----+-----+-----+-----+
+-----+-----+-----+-----+
|                NAME                |  SIZE  | STORAGE CLASS |
PROTOCOL |                BACKEND UUID         |  STATE  |  MANAGED  |
+-----+-----+-----+-----+
+-----+-----+-----+-----+
| pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7 | 1.0 GiB | ontapnas      |
file      | c5a6f6a4-b052-423b-80d4-8fb491a14a22 | online | true      |
+-----+-----+-----+-----+
+-----+-----+-----+-----+

```

Volumes importieren

Sie können vorhandene Storage Volumes mit als Kubernetes PV importieren
tridentctl import.

Überblick und Überlegungen

Ein Volume kann in Astra Trident importiert werden, um:

- Containerisierung einer Applikation und Wiederverwendung des vorhandenen Datensatzes
- Verwenden Sie einen Klon eines Datensatzes für eine kurzlebige Applikation
- Wiederherstellung eines fehlerhaften Kubernetes-Clusters
- Migration von Applikationsdaten bei der Disaster Recovery

Überlegungen

Lesen Sie vor dem Importieren eines Volumes die folgenden Überlegungen durch.

- Astra Trident kann nur ONTAP Volumes vom Typ RW (Lese-/Schreibzugriff) importieren. Volumes im DP-Typ (Datensicherung) sind SnapMirror Ziel-Volumes. Sie sollten die Spiegelungsbeziehung unterbrechen, bevor Sie das Volume in Astra Trident importieren.

- Wir empfehlen, Volumes ohne aktive Verbindungen zu importieren. Um ein aktiv verwendetes Volume zu importieren, klonen Sie das Volume, und führen Sie dann den Import durch.



Dies ist besonders für Block-Volumes wichtig, da Kubernetes die vorherige Verbindung nicht mitbekommt und problemlos ein aktives Volume an einen Pod anbinden kann. Dies kann zu Datenbeschädigungen führen.

- Aber `StorageClass` Muss auf einer PVC angegeben werden, Astra Trident verwendet diesen Parameter während des Imports nicht. Während der Volume-Erstellung werden Storage-Klassen eingesetzt, um basierend auf den Storage-Merkmalen aus verfügbaren Pools auszuwählen. Da das Volume bereits vorhanden ist, ist beim Import keine Poolauswahl erforderlich. Daher schlägt der Import auch dann nicht fehl, wenn das Volume auf einem Back-End oder Pool vorhanden ist, das nicht mit der in der PVC angegebenen Speicherklasse übereinstimmt.
- Die vorhandene Volumegröße wird in der PVC ermittelt und festgelegt. Nachdem das Volumen vom Speichertreiber importiert wurde, wird das PV mit einem ClaimRef an die PVC erzeugt.
 - Die Rückgewinnungsrichtlinie ist zunächst auf festgelegt `retain` Im PV. Nachdem Kubernetes die PVC und das PV erfolgreich bindet, wird die Zurückgewinnungsrichtlinie aktualisiert und an die Zurückgewinnungsrichtlinie der Storage-Klasse angepasst.
 - Wenn die Richtlinie zur Zurückgewinnung der Storage-Klasse lautet `delete`, Das Speichervolumen wird gelöscht, wenn das PV gelöscht wird.
- Astra Trident verwaltet standardmäßig die PVC und benennt die FlexVol und die LUN auf dem Backend um. Sie können die passieren `--no-manage` Flag zum Importieren eines nicht verwalteten Volumes. Wenn Sie verwenden `--no-manage`, Astra Trident führt keine zusätzlichen Operationen auf der PVC oder PV für den Lebenszyklus der Objekte. Das Speicher-Volume wird nicht gelöscht, wenn das PV gelöscht wird und andere Vorgänge wie Volume-Klon und Volume-Größe ebenfalls ignoriert werden.



Diese Option ist nützlich, wenn Sie Kubernetes für Workloads in Containern verwenden möchten, aber ansonsten den Lebenszyklus des Storage Volumes außerhalb von Kubernetes managen möchten.

- Der PVC und dem PV wird eine Anmerkung hinzugefügt, die einem doppelten Zweck dient, anzugeben, dass das Volumen importiert wurde und ob PVC und PV verwaltet werden. Diese Anmerkung darf nicht geändert oder entfernt werden.

Importieren Sie ein Volume

Verwenden Sie können `tridentctl import` Um ein Volume zu importieren.

Schritte

1. Erstellen der PVC-Datei (Persistent Volume Claim) (beispielsweise `pvc.yaml`), die verwendet werden, um die PVC zu erstellen. Die PVC-Datei sollte enthalten `name`, `namespace`, `accessModes`, und `storageClassName`. Optional können Sie angeben `unixPermissions` In Ihrer PVC-Definition.

Im Folgenden finden Sie ein Beispiel für eine Mindestspezifikation:

```

kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: my_claim
  namespace: my_namespace
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: my_storage_class

```



Verwenden Sie keine zusätzlichen Parameter wie den PV-Namen oder die Volume-Größe. Dies kann dazu führen, dass der Importbefehl fehlschlägt.

2. Verwenden Sie die `tridentctl import volume` Befehl zur Angabe des Namens des Astra Trident Back-End, das das Volume enthält, sowie des Namens, der das Volume auf dem Storage eindeutig identifiziert (z. B. ONTAP FlexVol, Element Volume, Cloud Volumes Service-Pfad). Der `-f` Argument ist erforderlich, um den Pfad zur PVC-Datei anzugeben.

```

tridentctl import volume <backendName> <volumeName> -f <path-to-pvc-
file>

```

Beispiele

Lesen Sie die folgenden Beispiele für den Import von Volumes für unterstützte Treiber.

ONTAP NAS und ONTAP NAS FlexGroup

Astra Trident unterstützt den Volume-Import mithilfe von `ontap-nas` Und `ontap-nas-flexgroup` Treiber.



- Der `ontap-nas-economy` Der Treiber kann qtrees nicht importieren und verwalten.
- Der `ontap-nas` Und `ontap-nas-flexgroup` Treiber erlauben keine doppelten Volume-Namen.

Jedes Volume wurde mit erstellt `ontap-nas` Treiber ist ein FlexVol auf dem ONTAP Cluster. Importieren von FlexVols mit dem `ontap-nas` Der Treiber funktioniert genauso. Eine FlexVol, die bereits auf einem ONTAP Cluster vorhanden ist, kann als importiert werden `ontap-nas` PVC: Ebenso können FlexGroup Volumes importiert werden als `ontap-nas-flexgroup` VES.

Beispiele für ONTAP NAS

Die folgende Darstellung zeigt ein Beispiel für ein verwaltetes Volume und einen nicht verwalteten Volume-Import.

Gemanagtes Volume

Im folgenden Beispiel wird ein Volume mit dem Namen importiert `managed_volume` Auf einem Backend mit dem Namen `ontap_nas`:

```
tridentctl import volume ontap_nas managed_volume -f <path-to-pvc-file>
```

| PROTOCOL | NAME | BACKEND UUID | SIZE | STORAGE CLASS | STATE | MANAGED |
|----------|--|--------------------------------------|---------|---------------|--------|---------|
| file | pvc-bf5ad463-afbb-11e9-8d9f-5254004dfdb7 | c5a6f6a4-b052-423b-80d4-8fb491a14a22 | 1.0 GiB | standard | online | true |

Nicht verwaltetes Volume

Bei Verwendung des `--no-manage` Argument, Astra Trident benennt das Volume nicht um.

Das folgende Beispiel importiert `unmanaged_volume` Auf dem `ontap_nas` Back-End:

```
tridentctl import volume nas_blog unmanaged_volume -f <path-to-pvc-file> --no-manage
```

| PROTOCOL | NAME | BACKEND UUID | SIZE | STORAGE CLASS | STATE | MANAGED |
|----------|--|--------------------------------------|---------|---------------|--------|---------|
| file | pvc-df07d542-afbc-11e9-8d9f-5254004dfdb7 | c5a6f6a4-b052-423b-80d4-8fb491a14a22 | 1.0 GiB | standard | online | false |

ONTAP SAN

Astra Trident unterstützt den Volume-Import mithilfe von `ontap-san` Treiber. Der Import von Volumes wird nicht unterstützt `ontap-san-economy` Treiber.

Astra Trident kann ONTAP SAN FlexVols importieren, die eine einzige LUN enthalten. Dies entspricht dem `ontap-san` Treiber, der für jede PVC und eine LUN innerhalb der FlexVol eine FlexVol erstellt. Astra Trident importiert die FlexVol und ordnet sie der PVC-Definition zu.

Beispiele für ONTAP SAN

Die folgende Darstellung zeigt ein Beispiel für ein verwaltetes Volume und einen nicht verwalteten Volume-Import.

Gemanagtes Volume

Für gemanagte Volumes benennt Astra Trident die FlexVol in den um `pvc-<uuid>` Formatieren Sie und die LUN innerhalb der FlexVol bis `lun0`.

Im folgenden Beispiel wird der importiert `ontap-san-managed` FlexVol, die auf dem vorhanden ist `ontap_san_default` Back-End:

```
tridentctl import volume ontapsan_san_default ontap-san-managed -f pvc-
basic-import.yaml -n trident -d
```

| PROTOCOL | NAME | BACKEND UUID | SIZE | STATE | STORAGE CLASS | MANAGED |
|----------|--|--------------------------------------|--------|--------|---------------|---------|
| block | pvc-d6ee4f54-4e40-4454-92fd-d00fc228d74a | cd394786-ddd5-4470-adc3-10c5ce4ca757 | 20 MiB | online | basic | true |

Nicht verwaltetes Volume

Das folgende Beispiel importiert `unmanaged_example_volume` Auf dem `ontap_san` Back-End:

```
tridentctl import volume -n trident san_blog unmanaged_example_volume
-f pvc-import.yaml --no-manage
```

| PROTOCOL | NAME | BACKEND UUID | SIZE | STATE | STORAGE CLASS | MANAGED |
|----------|--|--------------------------------------|---------|--------|---------------|---------|
| block | pvc-1fc999c9-ce8c-459c-82e4-ed4380a4b228 | e3275890-7d80-4af6-90cc-c7a0759f555a | 1.0 GiB | online | san-blog | false |

Wenn LUNS Initiatorgruppen zugeordnet sind, die einen IQN mit einem Kubernetes-Node-IQN teilen, wie im folgenden Beispiel dargestellt, erhalten Sie die Fehlermeldung: `LUN already mapped to initiator(s) in this group`. Sie müssen den Initiator entfernen oder die Zuordnung der LUN aufheben, um das Volume

zu importieren.

| Vserver | Igroup | Protocol | OS Type | Initiators |
|---------|---|----------|---------|------------------------------------|
| svm0 | k8s-nodename.example.com-fe5d36f2-cded-4f38-9eb0-c7719fc2f9f3 | iscsi | linux | iqn.1994-05.com.redhat:4c2e1cf35e0 |
| svm0 | unmanaged-example-igroup | mixed | linux | iqn.1994-05.com.redhat:4c2e1cf35e0 |

Element

Astra Trident unterstützt die NetApp Element Software und den NetApp HCI Volume-Import über die `solidfire-san` Treiber.



Der Elementtreiber unterstützt doppelte Volume-Namen. Astra Trident gibt jedoch einen Fehler zurück, wenn es doppelte Volume-Namen gibt. Um dies zu umgehen, klonen Sie das Volume, geben Sie einen eindeutigen Volume-Namen ein und importieren Sie das geklonte Volume.

Beispiel für ein Element

Im folgenden Beispiel wird ein importiert `element-managed` Volume am Backend `element_default`.

```
tridentctl import volume element_default element-managed -f pvc-basic-import.yaml -n trident -d
```

```
+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+
|          NAME          | SIZE | STORAGE CLASS |
PROTOCOL |          BACKEND UUID          | STATE | MANAGED |
+-----+-----+-----+-----+-----+-----+
| pvc-970ce1ca-2096-4ecd-8545-ac7edc24a8fe | 10 GiB | basic-element |
block   | d3ba047a-ea0b-43f9-9c42-e38e58301c49 | online | true   |
+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+
+-----+-----+-----+-----+-----+-----+
```

Google Cloud Platform

Astra Trident unterstützt den Volume-Import mithilfe von `gcp-cvs` Treiber.



Um ein Volume zu importieren, das von NetApp Cloud Volumes Service in die Google Cloud Platform unterstützt wird, identifizieren Sie das Volume anhand seines Volume-Pfads. Der Volume-Pfad ist der Teil des Exportpfades des Volumes nach dem :/. Beispiel: Wenn der Exportpfad lautet 10.0.0.1:/adroit-jolly-swift, Der Volume-Pfad ist adroit-jolly-swift.

Beispiel für die Google Cloud Platform

Im folgenden Beispiel wird ein importiert gcp-cvs Volume am Backend gcpcvs_YEppr Mit dem Volume-Pfad von adroit-jolly-swift.

```
tridentctl import volume gcpcvs_YEppr adroit-jolly-swift -f <path-to-pvc-
file> -n trident
```

| PROTOCOL | NAME | BACKEND UUID | SIZE | STORAGE CLASS | STATE | MANAGED |
|----------|--|--------------------------------------|--------|---------------|--------|---------|
| | pvc-a46ccab7-44aa-4433-94b1-e47fc8c0fa55 | e1a6e65b-299e-4568-ad05-4f0a105c888f | 93 GiB | gcp-storage | online | true |
| | | | | file | | |

Azure NetApp Dateien

Astra Trident unterstützt den Volume-Import mithilfe von azure-netapp-files Treiber.



Um ein Azure NetApp Files-Volume zu importieren, identifizieren Sie das Volume anhand seines Volume-Pfads. Der Volume-Pfad ist der Teil des Exportpfades des Volumes nach dem :/. Beispiel: Wenn der Mount-Pfad lautet 10.0.0.2:/importvoll, Der Volume-Pfad ist importvoll.

Beispiel: Azure NetApp Files

Im folgenden Beispiel wird ein importiert azure-netapp-files Volume am Backend azurenetappfiles_40517 Mit dem Volume-Pfad importvoll.

```
tridentctl import volume azurenetappfiles_40517 importvoll -f <path-to-pvc-file> -n trident
```

```
+-----+-----+-----+
+-----+-----+-----+-----+
|          NAME          | SIZE  | STORAGE CLASS |
PROTOCOL |          BACKEND UUID          | STATE | MANAGED |
+-----+-----+-----+
+-----+-----+-----+-----+
| pvc-0ee95d60-fd5c-448d-b505-b72901b3a4ab | 100 GiB | anf-storage  |
file      | 1c01274f-d94b-44a3-98a3-04c953c9a51e | online | true      |
+-----+-----+-----+
+-----+-----+-----+-----+
```

Ein NFS-Volume kann über Namespaces hinweg genutzt werden

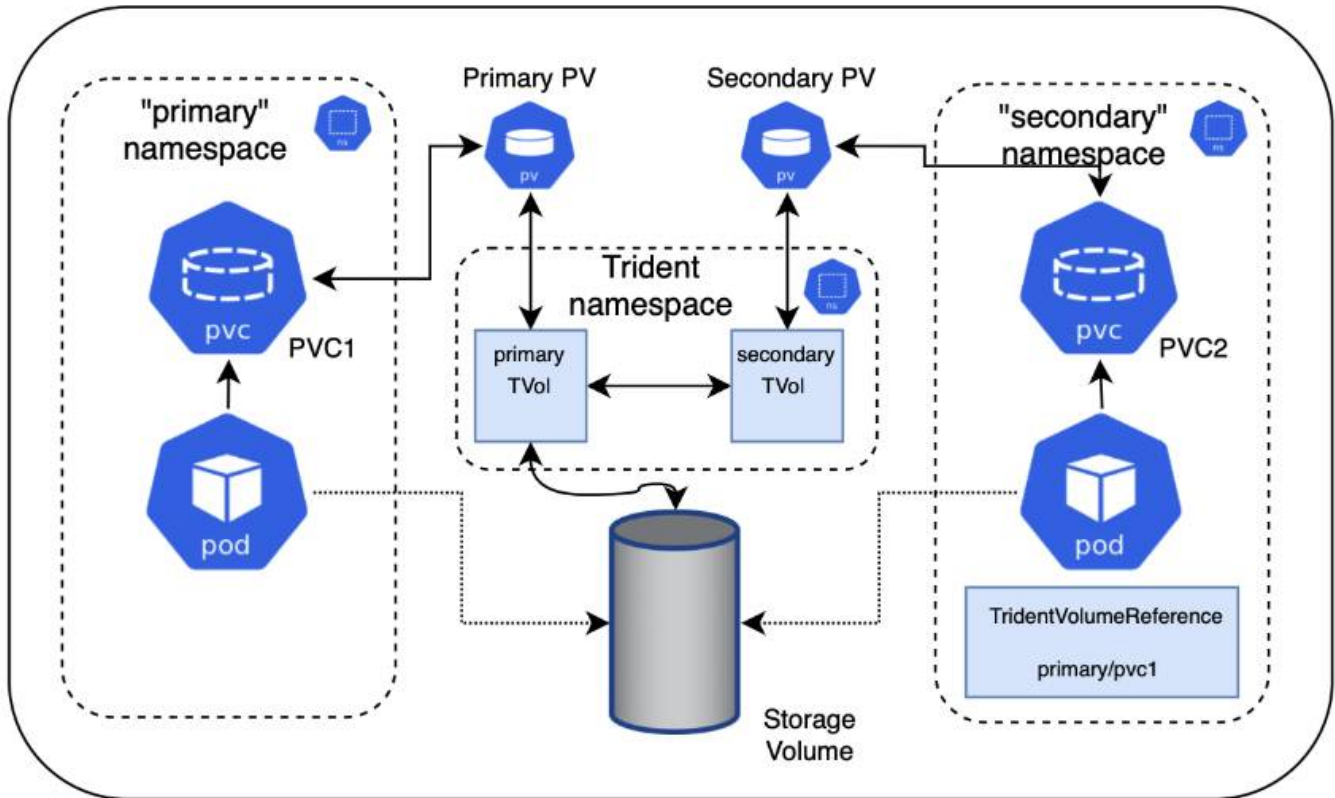
Mit Astra Trident können Sie ein Volume in einem primären Namespace erstellen und es in einem oder mehreren sekundären Namespaces teilen.

Funktionen

Mit dem Astra TridentVolumeReference CR können Sie ReadWriteManche (RWX) NFS-Volumes sicher über einen oder mehrere Kubernetes-Namespaces teilen. Diese native Kubernetes-Lösung bietet folgende Vorteile:

- Mehrere Stufen der Zugriffssteuerung zur Sicherstellung der Sicherheit
- Funktioniert mit allen Trident NFS-Volume-Treibern
- Tridentctl oder andere nicht-native Kubernetes-Funktionen sind nicht von Bedeutung

Dieses Diagramm zeigt die NFS-Volume-Freigabe über zwei Kubernetes-Namespaces.



Schnellstart

Sie können in nur wenigen Schritten NFS-Volume Sharing einrichten.

1

Konfigurieren Sie die PVC-Quelle für die gemeinsame Nutzung des Volumes

Der Eigentümer des Quell-Namespace erteilt die Berechtigung, auf die Daten im Quell-PVC zuzugreifen.

2

Berechtigung zum Erstellen eines CR im Ziel-Namespace gewähren

Der Clusteradministrator erteilt dem Eigentümer des Ziel-Namespace die Berechtigung, das TridentVolumeReference CR zu erstellen.

3

Erstellen Sie im Ziel-Namespace tridentVolumeReference

Der Eigentümer des Ziel-Namespace erstellt das TridentVolumeReference CR, um sich auf das Quell-PVC zu beziehen.

4

Erstellen Sie das untergeordnete PVC im Ziel-Namespace

Der Eigentümer des Ziel-Namespace erstellt das untergeordnete PVC, um die Datenquelle aus dem Quell-PVC zu verwenden.

Konfigurieren Sie die Namensräume für Quelle und Ziel

Um die Sicherheit zu gewährleisten, erfordert die Namespace-übergreifende Freigabe Zusammenarbeit und Aktion durch den Eigentümer des Quell-Namespace, den Cluster-Administrator und den Ziel-Namespace-Eigentümer. In jedem Schritt wird die Benutzerrolle festgelegt.

Schritte

1. **Source Namespace Owner:** Erstellen Sie das PVC (`pvc1`) Im Quell-Namespace, der die Erlaubnis gibt, mit dem Ziel-Namespace zu teilen (`namespace2`) Mit dem `shareToNamespace` Anmerkung:

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc1
  namespace: namespace1
  annotations:
    trident.netapp.io/shareToNamespace: namespace2
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: trident-csi
  resources:
    requests:
      storage: 100Gi
```

Astra Trident erstellt das PV und das Back-End NFS Storage Volume.



- Sie können das PVC über eine durch Kommas getrennte Liste mehreren Namespaces freigeben. Beispiel: `trident.netapp.io/shareToNamespace: namespace2, namespace3, namespace4`.
- Sie können mit allen Namespaces freigeben *. Beispiel: `trident.netapp.io/shareToNamespace: *`
- Sie können das PVC so aktualisieren, dass es die enthält `shareToNamespace` Kommentare können jederzeit hinzugefügt werden.

2. **Cluster Admin:** Erstellen Sie die benutzerdefinierte Rolle und `kubeconfig`, um dem Ziel-Namespace-Eigentümer die Berechtigung zu erteilen, das `TridentVolumeReference` CR im Ziel-Namespace zu erstellen.
3. **Zielgebietes-Namespace-Eigentümer:** Erstellen Sie ein `TridentVolumeReference` CR im Ziel-Namespace, der sich auf den Quell-Namespace bezieht `pvc1`.

```

apiVersion: trident.netapp.io/v1
kind: TridentVolumeReference
metadata:
  name: my-first-tvr
  namespace: namespace2
spec:
  pvcName: pvc1
  pvcNamespace: namespace1

```

4. **Eigentümer des Ziel-Namespace:** Erstellen Sie ein PVC (`pvc2`) Im Ziel-Namespace (`namespace2`) Mit dem `shareFromPVC` Anmerkung zur Angabe der Quelle PVC.

```

kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  annotations:
    trident.netapp.io/shareFromPVC: namespace1/pvc1
  name: pvc2
  namespace: namespace2
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: trident-csi
  resources:
    requests:
      storage: 100Gi

```



Die Größe der Ziel-PVC muss kleiner oder gleich der Quelle PVC sein.

Ergebnisse

Astra Trident liest den `shareFromPVC` Anmerkung auf dem Ziel-PVC und erstellt das Ziel-PV als untergeordnetes Volumen ohne eigene Speicherressource, die auf das Quell-PV verweist und die PV-Quellressource teilt. Die Ziel-PVC und das PV erscheinen wie normal gebunden.

Löschen eines freigegebenen Volumes

Sie können ein Volume löschen, das über mehrere Namespaces hinweg gemeinsam genutzt wird. Astra Trident entfernt den Zugriff auf das Volume im Quell-Namespace und behält auch andere Namespaces, die das Volume gemeinsam nutzen. Wenn alle Namespaces entfernt werden, die auf dem Volume verweisen, löscht Astra Trident das Volume.

Nutzung `tridentctl get` Zum Abfragen von untergeordneten Volumes

Verwenden der `tridentctl` Das Dienstprogramm kann ausgeführt werden `get` Befehl zum Abrufen untergeordneter Volumes. Weitere Informationen finden Sie unter [Link:../Trident-](#)

Referenz/tridentctl.html

Usage:

```
tridentctl get [option]
```

Markierungen:

- `-h, --help`: Hilfe für Volumen.
- `--parentOfSubordinate string`: Abfrage auf untergeordnetes Quellvolumen begrenzen.
- `--subordinateOf string`: Abfrage auf Untergerbene beschränken.

Einschränkungen

- Astra Trident kann nicht verhindern, dass Ziel-Namespace auf dem Shared Volume schreiben. Sie sollten Dateisperren oder andere Prozesse verwenden, um das Überschreiben von gemeinsam genutzten Volume-Daten zu verhindern.
- Sie können den Zugriff auf die Quelle PVC nicht widerrufen, indem Sie die entfernen `shareToNamespace` Oder `shareFromNamespace` Anmerkungen oder Löschen des `TridentVolumeReference` CR. Um den Zugriff zu widerrufen, müssen Sie das untergeordnete PVC löschen.
- Snapshots, Klone und Spiegelungen sind auf untergeordneten Volumes nicht möglich.

Finden Sie weitere Informationen

Weitere Informationen zum Namespace-übergreifenden Volume-Zugriff:

- Besuchen Sie ["Teilen von Volumes zwischen Namespaces: Sagen Sie hallo für Namespace-übergreifenden Volume-Zugriff"](#).
- Demo ansehen am ["NetAppTV"](#).

Verwenden Sie die CSI-Topologie

Astra Trident kann Volumes selektiv erstellen und zu Nodes in einem Kubernetes Cluster verbinden, indem der verwendet wird ["Funktion CSI Topology"](#).

Überblick

Mithilfe der CSI Topology-Funktion kann der Zugriff auf Volumes auf einen Teil von Nodes basierend auf Regionen und Verfügbarkeitszonen begrenzt werden. Cloud-Provider ermöglichen Kubernetes-Administratoren inzwischen das Erstellen von Nodes, die zonenbasiert sind. Die Nodes können sich in verschiedenen Verfügbarkeitszonen innerhalb einer Region oder über verschiedene Regionen hinweg befinden. Astra Trident verwendet CSI Topology, um die Provisionierung von Volumes für Workloads in einer Multi-Zone-Architektur zu vereinfachen.



Erfahren Sie mehr über die Funktion CSI Topology ["Hier"](#).

Kubernetes bietet zwei unterschiedliche Modi für die Volume-Bindung:

- Mit `VolumeBindingMode` Auf einstellen `Immediate`, Astra Trident erstellt das Volume ohne

Topologiebewusstsein. Die Volume-Bindung und die dynamische Bereitstellung werden bei der Erstellung des PVC behandelt. Dies ist die Standardeinstellung `VolumeBindingMode` Und ist für Cluster geeignet, die keine Topologiebeschränkungen mehr durchsetzen. Persistente Volumes werden erstellt, ohne von den Planungsanforderungen des anfragenden Pods abhängig zu sein.

- Mit `VolumeBindingMode` Auf einstellen `WaitForFirstConsumer`, Die Erstellung und Bindung eines Persistent Volume für ein PVC wird verzögert, bis ein Pod, der die PVC verwendet, geplant und erstellt wird. Auf diese Weise werden Volumes erstellt, um Planungseinschränkungen zu erfüllen, die durch Topologieanforderungen durchgesetzt werden.



Der `WaitForFirstConsumer` Für den Bindungsmodus sind keine Topologiebeschriftungen erforderlich. Diese kann unabhängig von der CSI Topology Funktion verwendet werden.

Was Sie benötigen

Für die Verwendung von CSI Topology benötigen Sie Folgendes:

- Einen Kubernetes-Cluster mit einem "[Unterstützte Kubernetes-Version](#)"

```
kubectl version
Client Version: version.Info{Major:"1", Minor:"19",
GitVersion:"v1.19.3",
GitCommit:"1e11e4a2108024935ecfcb2912226cedeafd99df",
GitTreeState:"clean", BuildDate:"2020-10-14T12:50:19Z",
GoVersion:"go1.15.2", Compiler:"gc", Platform:"linux/amd64"}
Server Version: version.Info{Major:"1", Minor:"19",
GitVersion:"v1.19.3",
GitCommit:"1e11e4a2108024935ecfcb2912226cedeafd99df",
GitTreeState:"clean", BuildDate:"2020-10-14T12:41:49Z",
GoVersion:"go1.15.2", Compiler:"gc", Platform:"linux/amd64"}
```

- Nodes im Cluster sollten über Labels verfügen, die eine Topologiebewusstsein einführen (`topology.kubernetes.io/region` Und `topology.kubernetes.io/zone`). Diese Labels * sollten auf Knoten im Cluster vorhanden sein* bevor Astra Trident installiert ist, damit Astra Trident Topologieorientiert ist.

```
kubectl get nodes -o=jsonpath='{range .items[*]}[.metadata.name],
{.metadata.labels}]{"\n"}{end}' | grep --color "topology.kubernetes.io"
[node1,
{"beta.kubernetes.io/arch":"amd64","beta.kubernetes.io/os":"linux","kuber-
netes.io/arch":"amd64","kubernetes.io/hostname":"node1","kubernetes.io/
os":"linux","node-
role.kubernetes.io/master":"","topology.kubernetes.io/region":"us-
east1","topology.kubernetes.io/zone":"us-east1-a"}]
[node2,
{"beta.kubernetes.io/arch":"amd64","beta.kubernetes.io/os":"linux","kuber-
netes.io/arch":"amd64","kubernetes.io/hostname":"node2","kubernetes.io/
os":"linux","node-
role.kubernetes.io/worker":"","topology.kubernetes.io/region":"us-
east1","topology.kubernetes.io/zone":"us-east1-b"}]
[node3,
{"beta.kubernetes.io/arch":"amd64","beta.kubernetes.io/os":"linux","kuber-
netes.io/arch":"amd64","kubernetes.io/hostname":"node3","kubernetes.io/
os":"linux","node-
role.kubernetes.io/worker":"","topology.kubernetes.io/region":"us-
east1","topology.kubernetes.io/zone":"us-east1-c"}]
```

Schritt 1: Erstellen Sie ein Topologieorientiertes Backend

Astra Trident Storage-Back-Ends können für die selektive Bereitstellung von Volumes basierend auf Verfügbarkeitszonen ausgelegt werden. Jedes Backend kann optional mittragen `supportedTopologies` Block, der eine Liste der zu unterstützenden Zonen und Regionen darstellt. Bei `StorageClasses`, die ein solches Backend nutzen, wird ein Volume nur erstellt, wenn es von einer Applikation angefordert wird, die in einer unterstützten Region/Zone geplant ist.

Hier ist eine Beispiel-Backend-Definition:

YAML

```
---
version: 1
storageDriverName: ontap-san
backendName: san-backend-us-east1
managementLIF: 192.168.27.5
svm: iscsi_svm
username: admin
password: password
supportedTopologies:
- topology.kubernetes.io/region: us-east1
  topology.kubernetes.io/zone: us-east1-a
- topology.kubernetes.io/region: us-east1
  topology.kubernetes.io/zone: us-east1-b
```

JSON

```
{
  "version": 1,
  "storageDriverName": "ontap-san",
  "backendName": "san-backend-us-east1",
  "managementLIF": "192.168.27.5",
  "svm": "iscsi_svm",
  "username": "admin",
  "password": "password",
  "supportedTopologies": [
    {"topology.kubernetes.io/region": "us-east1",
     "topology.kubernetes.io/zone": "us-east1-a"},
    {"topology.kubernetes.io/region": "us-east1",
     "topology.kubernetes.io/zone": "us-east1-b"}
  ]
}
```



`supportedTopologies` Wird verwendet, um eine Liste von Regionen und Zonen pro Backend bereitzustellen. Diese Regionen und Zonen stellen die Liste der zulässigen Werte dar, die in einer StorageClass bereitgestellt werden können. Bei StorageClasses, die einen Teil der Regionen und Zonen enthalten, die in einem Backend bereitgestellt werden, erstellt Astra Trident ein Volume im Backend.

Sie können definieren `supportedTopologies` Auch pro Storagepool. Das folgende Beispiel zeigt:

```

---
version: 1
storageDriverName: ontap-nas
backendName: nas-backend-us-centrall
managementLIF: 172.16.238.5
svm: nfs_svm
username: admin
password: password
supportedTopologies:
- topology.kubernetes.io/region: us-centrall
  topology.kubernetes.io/zone: us-centrall-a
- topology.kubernetes.io/region: us-centrall
  topology.kubernetes.io/zone: us-centrall-b
storage:
- labels:
    workload: production
    region: Iowa-DC
    zone: Iowa-DC-A
    supportedTopologies:
    - topology.kubernetes.io/region: us-centrall
      topology.kubernetes.io/zone: us-centrall-a
- labels:
    workload: dev
    region: Iowa-DC
    zone: Iowa-DC-B
    supportedTopologies:
    - topology.kubernetes.io/region: us-centrall
      topology.kubernetes.io/zone: us-centrall-b

```

In diesem Beispiel ist der `region` Und `zone` Etiketten stehen für die Position des Speicherpools. `topology.kubernetes.io/region` Und `topology.kubernetes.io/zone` Vorgeben, woher die Speicherpools verbraucht werden können.

Schritt: Definition von StorageClasses, die sich der Topologie bewusst sind

Auf der Grundlage der Topologiebeschriftungen, die den Nodes im Cluster zur Verfügung gestellt werden, können StorageClasses so definiert werden, dass sie Topologieinformationen enthalten. So werden die Storage-Pools festgelegt, die als Kandidaten für PVC-Anfragen dienen, und die Untergruppe der Nodes, die die von Trident bereitgestellten Volumes nutzen können.

Das folgende Beispiel zeigt:

```

apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
name: netapp-san-us-east1
provisioner: csi.trident.netapp.io
volumeBindingMode: WaitForFirstConsumer
allowedTopologies:
- matchLabelExpressions:
- key: topology.kubernetes.io/zone
  values:
  - us-east1-a
  - us-east1-b
- key: topology.kubernetes.io/region
  values:
  - us-east1
parameters:
  fsType: "ext4"

```

In der oben angegebenen StorageClass-Definition `volumeBindingMode` ist auf festgelegt `WaitForFirstConsumer`. VES, die mit dieser StorageClass angefordert werden, werden erst dann gehandelt, wenn sie in einem Pod referenziert werden. Und `allowedTopologies` stellt die Zonen und die Region bereit, die verwendet werden sollen. Der `netapp-san-us-east1` StorageClass erstellt VES auf dem `san-backend-us-east1` Back-End oben definiert.

Schritt 3: Erstellen und verwenden Sie ein PVC

Wenn die StorageClass erstellt und einem Backend zugeordnet wird, können Sie jetzt PVCs erstellen.

Siehe Beispiel `spec` Unten:

```

---
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
name: pvc-san
spec:
accessModes:
  - ReadWriteOnce
resources:
  requests:
    storage: 300Mi
storageClassName: netapp-san-us-east1

```

Das Erstellen eines PVC mithilfe dieses Manifests würde Folgendes zur Folge haben:


```

kubect1 create -f pvc.yaml
persistentvolumeclaim/pvc-san created
kubect1 get pvc
NAME          STATUS      VOLUME      CAPACITY    ACCESS MODES    STORAGECLASS
AGE
pvc-san      Pending
2s
kubect1 describe pvc
Name:          pvc-san
Namespace:    default
StorageClass: netapp-san-us-east1
Status:       Pending
Volume:
Labels:       <none>
Annotations:  <none>
Finalizers:   [kubernetes.io/pvc-protection]
Capacity:
Access Modes:
VolumeMode:   Filesystem
Mounted By:   <none>
Events:
  Type      Reason              Age    From
  ----      -
  Normal    WaitForFirstConsumer 6s     persistentvolume-controller
waiting
for first consumer to be created before binding

```

Verwenden Sie für Trident, ein Volume zu erstellen und es an die PVC zu binden, das in einem Pod verwendet wird. Das folgende Beispiel zeigt:

```

apiVersion: v1
kind: Pod
metadata:
  name: app-pod-1
spec:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
          - matchExpressions:
              - key: topology.kubernetes.io/region
                operator: In
                values:
                  - us-east1
            preferredDuringSchedulingIgnoredDuringExecution:
              - weight: 1
                preference:
                  matchExpressions:
                    - key: topology.kubernetes.io/zone
                      operator: In
                      values:
                        - us-east1-a
                        - us-east1-b
      securityContext:
        runAsUser: 1000
        runAsGroup: 3000
        fsGroup: 2000
    volumes:
      - name: voll
        persistentVolumeClaim:
          claimName: pvc-san
    containers:
      - name: sec-ctx-demo
        image: busybox
        command: [ "sh", "-c", "sleep 1h" ]
        volumeMounts:
          - name: voll
            mountPath: /data/demo
        securityContext:
          allowPrivilegeEscalation: false

```

Diese PodSpec beauftragt Kubernetes, den Pod auf Nodes zu planen, die in vorhanden sind us-east1 Wählen Sie einen beliebigen Knoten aus, der im vorhanden ist us-east1-a Oder us-east1-b Zonen:

Siehe die folgende Ausgabe:

```
kubectl get pods -o wide
NAME          READY   STATUS    RESTARTS   AGE   IP              NODE
NOMINATED NODE READINESS GATES
app-pod-1    1/1     Running   0           19s   192.168.25.131  node2
<none>      <none>
kubectl get pvc -o wide
NAME          STATUS   VOLUME                                     CAPACITY
ACCESS MODES STORAGECLASS          AGE   VOLUMEMODE
pvc-san      Bound   pvc-ecb1e1a0-840c-463b-8b65-b3d033e2e62b 300Mi
RWO          netapp-san-us-east1  48s   Filesystem
```

Aktualisieren Sie Back-Ends, um einzuschließen `supportedTopologies`

Vorhandene Back-Ends können mit einer Liste von aktualisiert werden `supportedTopologies` Wird verwendet `tridentctl backend update`. Dies wirkt sich nicht auf Volumes aus, die bereits bereitgestellt wurden und nur für nachfolgende VES verwendet werden.

Weitere Informationen

- ["Management von Ressourcen für Container"](#)
- ["NodeSelector"](#)
- ["Affinität und Antiaffinität"](#)
- ["Tönungen und Tolerationen"](#)

Arbeiten Sie mit Snapshots

Kubernetes Volume Snapshots von Persistent Volumes (PVs) ermöglichen zeitpunktgenaue Kopien von Volumes. Sie können einen Snapshot eines mit Astra Trident erstellten Volumes erstellen, einen außerhalb von Astra Trident erstellten Snapshot importieren, ein neues Volume aus einem vorhandenen Snapshot erstellen und Volume-Daten aus Snapshots wiederherstellen.

Überblick

Volume Snapshot wird von unterstützt `ontap-nas`, `ontap-nas-flexgroup`, `ontap-san`, `ontap-san-economy`, `solidfire-san`, `gcp-cvs`, und `azure-netapp-files` Treiber.

Bevor Sie beginnen

Sie benötigen einen externen Snapshot-Controller und benutzerdefinierte Ressourcendefinitionen (CRDs), um mit Snapshots arbeiten zu können. Dies ist die Aufgabe des Kubernetes Orchestrator (z. B. Kubeadm, GKE, OpenShift).

Wenn die Kubernetes-Distribution den Snapshot-Controller und die CRDs nicht enthält, lesen Sie [Stellen Sie einen Volume-Snapshot-Controller bereit](#).



Erstellen Sie keinen Snapshot Controller, wenn Sie On-Demand Volume Snapshots in einer GKE-Umgebung erstellen. GKE verwendet einen integrierten, versteckten Snapshot-Controller.

Erstellen eines Volume-Snapshots

Schritte

1. Erstellen Sie ein `VolumeSnapshotClass`. Weitere Informationen finden Sie unter "[VolumeSnapshotKlasse](#)".
 - Der `driver` Verweist auf den Astra Trident CSI-Treiber.
 - `deletionPolicy` Kann sein `Delete` Oder `Retain`. Wenn eingestellt auf `Retain`, Der zugrunde liegende physische Snapshot auf dem Storage-Cluster wird auch dann beibehalten, wenn der `VolumeSnapshot` Objekt wurde gelöscht.

Beispiel

```
cat snap-sc.yaml
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshotClass
metadata:
  name: csi-snapclass
driver: csi.trident.netapp.io
deletionPolicy: Delete
```

2. Erstellen Sie einen Snapshot einer vorhandenen PVC.

Beispiele

- In diesem Beispiel wird ein Snapshot eines vorhandenen PVC erstellt.

```
cat snap.yaml
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshot
metadata:
  name: pvc1-snap
spec:
  volumeSnapshotClassName: csi-snapclass
  source:
    persistentVolumeClaimName: pvc1
```

- In diesem Beispiel wird ein Volume-Snapshot-Objekt für eine PVC mit dem Namen erstellt `pvc1` Der Name des Snapshots lautet `pvc1-snap`. Ein `VolumeSnapshot` ist analog zu einem PVC und einem zugeordnet `VolumeSnapshotContent` Objekt, das den tatsächlichen Snapshot darstellt.

```
kubectl create -f snap.yaml
volumesnapshot.snapshot.storage.k8s.io/pvc1-snap created

kubectl get volumesnapshots
NAME                AGE
pvc1-snap           50s
```

- Sie können den identifizieren `VolumeSnapshotContent` Objekt für das `pvc1-snap` `VolumeSnapshot` wird beschrieben. Der `Snapshot Content Name` Identifiziert das `VolumeSnapshotContent`-Objekt, das diesen Snapshot bereitstellt. Der `Ready To Use` Parameter gibt an, dass der Snapshot zum Erstellen einer neuen PVC verwendet werden kann.

```
kubectl describe volumesnapshots pvc1-snap
Name:          pvc1-snap
Namespace:     default
.
.
.
Spec:
  Snapshot Class Name:  pvc1-snap
  Snapshot Content Name: snapcontent-e8d8a0ca-9826-11e9-9807-
525400f3f660
  Source:
    API Group:
    Kind:      PersistentVolumeClaim
    Name:      pvc1
Status:
  Creation Time:  2019-06-26T15:27:29Z
  Ready To Use:  true
  Restore Size:  3Gi
.
.
```

Erstellen Sie eine PVC aus einem Volume-Snapshot

Verwenden Sie können `dataSource` So erstellen Sie eine PVC mit einem `VolumeSnapshot` namens `<pvc-name>` Als Quelle der Daten. Nachdem die PVC erstellt wurde, kann sie an einem Pod befestigt und wie jedes andere PVC verwendet werden.



Die PVC wird im selben Backend wie das Quell-Volumen erstellt. Siehe "[KB: Die Erstellung einer PVC aus einem Trident PVC-Snapshot kann nicht in einem alternativen Backend erstellt werden](#)".

Im folgenden Beispiel wird die PVC mit erstellt `pvc1-snap` Als Datenquelle gespeichert.

```

cat pvc-from-snap.yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-from-snap
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: golden
  resources:
    requests:
      storage: 3Gi
  dataSource:
    name: pvcl-snap
    kind: VolumeSnapshot
    apiGroup: snapshot.storage.k8s.io

```

Importieren Sie einen Volume-Snapshot

Astra Trident unterstützt das ["Vorab bereitgestellter Snapshot-Prozess von Kubernetes"](#) Damit der Clusteradministrator einen erstellen kann `VolumeSnapshotContent` Objekt- und Import von Snapshots, die außerhalb von Astra Trident erstellt wurden.

Bevor Sie beginnen

Astra Trident muss das übergeordnete Volume des Snapshots erstellt oder importiert haben.

Schritte

1. **Cluster admin:** Erstellen Sie eine `VolumeSnapshotContent` Objekt, das auf den Back-End-Snapshot verweist. Dadurch wird der Snapshot Workflow in Astra Trident gestartet.
 - Geben Sie den Namen des Back-End-Snapshots in an annotations Als `trident.netapp.io/internalSnapshotName: <"backend-snapshot-name">`.
 - Angeben `<name-of-parent-volume-in-trident>/<volume-snapshot-content-name>` In `snapshotHandle`. Dies ist die einzige Information, die Astra Trident vom externen Snapshot in zur Verfügung gestellt wird `ListSnapshots` Anruf.



Der `<volumeSnapshotContentName>` Aufgrund von Einschränkungen bei der CR-Benennung kann der Name des Back-End-Snapshots nicht immer übereinstimmen.

Beispiel

Im folgenden Beispiel wird ein erstellt `VolumeSnapshotContent` Objekt, das auf Back-End-Snapshot verweist `snap-01`.

```

apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshotContent
metadata:
  name: import-snap-content
  annotations:
    trident.netapp.io/internalSnapshotName: "snap-01" # This is the
name of the snapshot on the backend
spec:
  deletionPolicy: Retain
  driver: csi.trident.netapp.io
  source:
    snapshotHandle: pvc-f71223b5-23b9-4235-bbfe-e269ac7b84b0/import-
snap-content # <import PV name or source PV name>/<volume-snapshot-
content-name>

```

2. **Cluster admin:** Erstellen Sie das `VolumeSnapshot` CR, der auf den `VolumeSnapshotContent` Objekt: Dadurch wird der Zugriff auf die Verwendung des angefordert `VolumeSnapshot` In einem bestimmten Namespace.

Beispiel

Im folgenden Beispiel wird ein erstellt `VolumeSnapshot` CR benannt `import-snap` Die auf die `VolumeSnapshotContent` Genannt `import-snap-content`.

```

apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshot
metadata:
  name: import-snap
spec:
  # volumeSnapshotClassName: csi-snapclass (not required for pre-
provisioned or imported snapshots)
  source:
    volumeSnapshotContentName: import-snap-content

```

3. **Interne Verarbeitung (keine Aktion erforderlich):** der externe Snapshotter erkennt das neu erstellte `VolumeSnapshotContent` Und führt das aus `ListSnapshots` Anruf. Astra Trident erstellt die `TridentSnapshot`.
- Der externe Schnapper legt den fest `VolumeSnapshotContent` Bis `readyToUse` Und das `VolumeSnapshot` Bis `true`.
 - Trident kehrt zurück `readyToUse=true`.
4. **Jeder Benutzer:** Erstellen Sie eine `PersistentVolumeClaim` Um auf das neue zu verweisen `VolumeSnapshot`, Wo der `spec.dataSource` (Oder `spec.dataSourceRef`) Name ist der `VolumeSnapshot` Name:

Beispiel

Im folgenden Beispiel wird eine PVC erstellt, die auf den verweist VolumeSnapshot Genannt import-snap.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-from-snap
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: simple-sc
  resources:
    requests:
      storage: 1Gi
  dataSource:
    name: import-snap
    kind: VolumeSnapshot
    apiGroup: snapshot.storage.k8s.io
```

Stellen Sie Volume-Daten mithilfe von Snapshots wieder her

Das Snapshot-Verzeichnis ist standardmäßig ausgeblendet, um die maximale Kompatibilität von Volumes zu ermöglichen, die über bereitgestellt werden `ontap-nas` Und `ontap-nas-economy` Treiber. Aktivieren Sie die `.snapshot` Verzeichnis, um Daten von Snapshots direkt wiederherzustellen.

Verwenden Sie die ONTAP-CLI zur Wiederherstellung eines Volume-Snapshots, um einen in einem früheren Snapshot aufgezeichneten Zustand wiederherzustellen.

```
cluster1::*> volume snapshot restore -vserver vs0 -volume vol3 -snapshot
vol3_snap_archive
```



Wenn Sie eine Snapshot-Kopie wiederherstellen, wird die vorhandene Volume-Konfiguration überschrieben. Änderungen an den Volume-Daten nach der Erstellung der Snapshot Kopie gehen verloren.

Löschen Sie ein PV mit den zugehörigen Snapshots

Wenn Sie ein persistentes Volume mit zugeordneten Snapshots löschen, wird das entsprechende Trident-Volume in einen „Löschzustand“ aktualisiert. Entfernen Sie die Volume Snapshots, um das Astra Trident Volume zu löschen.

Stellen Sie einen Volume-Snapshot-Controller bereit

Wenn Ihre Kubernetes-Distribution den Snapshot-Controller und CRDs nicht enthält, können Sie sie wie folgt bereitstellen.

Schritte

1. Erstellen von Volume Snapshot-CRDs.

```
cat snapshot-setup.sh
#!/bin/bash
# Create volume snapshot CRDs
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
6.1/client/config/crd/snapshot.storage.k8s.io_volumesnapshotclasses.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
6.1/client/config/crd/snapshot.storage.k8s.io_volumesnapshotcontents.yam
l
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
6.1/client/config/crd/snapshot.storage.k8s.io_volumesnapshots.yaml
```

2. Erstellen Sie den Snapshot-Controller.

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-6.1/deploy/kubernetes/snapshot-
controller/rbac-snapshot-controller.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-6.1/deploy/kubernetes/snapshot-
controller/setup-snapshot-controller.yaml
```



Öffnen Sie bei Bedarf `deploy/kubernetes/snapshot-controller/rbac-snapshot-controller.yaml` Und Aktualisierung namespace In Ihren Namespace.

Weiterführende Links

- ["Volume Snapshots"](#)
- ["VolumeSnapshotKlasse"](#)

Management und Monitoring von Astra Trident

Upgrade Astra Trident

Upgrade Astra Trident

Astra Trident folgt einem vierteljährlichen Release-Intervall mit vier Hauptversionen pro Kalenderjahr. Jede neue Version baut auf den vorherigen Versionen auf und bietet neue Funktionen, Performance-Verbesserungen, Bug Fixes und Verbesserungen. Wir empfehlen Ihnen, ein Upgrade mindestens einmal pro Jahr durchzuführen, um von den neuen Funktionen in Astra Trident zu profitieren.

Überlegungen vor dem Upgrade

Bei einem Upgrade auf die neueste Version von Astra Trident sollten Sie Folgendes berücksichtigen:

- In allen Namespaces in einem Kubernetes-Cluster sollte nur eine Astra Trident Instanz installiert werden.
- Astra Trident 23.07 und höher benötigt v1-Volume-Snapshots und unterstützt keine Alpha- oder Beta-Snapshots mehr.
- Wenn Sie Cloud Volumes Service für Google Cloud in erstellt haben "[CVS-Diensttyp](#)", Sie müssen die Backend-Konfiguration aktualisieren, um die zu verwenden `standardsw` Oder `zoneredundantstandardsw` Service-Level beim Upgrade von Astra Trident 23.01. Fehler beim Aktualisieren des `serviceLevel` Im Backend kann zu einem Ausfall von Volumes führen. Siehe "[Beispiele für CVS-Diensttypen](#)" Entsprechende Details.
- Beim Upgrade ist es wichtig, dass Sie das Upgrade durchführen `parameter.fsType` In `StorageClasses` Verwendet von Astra Trident. Sie können löschen und neu erstellen `StorageClasses` Ohne Unterbrechung vorhandener Volumes
 - Dies ist eine **Anforderung** für die Durchsetzung "[Sicherheitskontexte](#)" Für SAN-Volumes.
 - Das Verzeichnis [sample input](#) enthält Beispiele wie `storage-class-basic.yaml.templ` und Link:[https://github.com/NetApp/trident/blob/master/trident-installer/sample-input/storage-class-samples/storage-class-bronze-default.yaml\[storage-class-bronze-default.yaml^\]](https://github.com/NetApp/trident/blob/master/trident-installer/sample-input/storage-class-samples/storage-class-bronze-default.yaml[storage-class-bronze-default.yaml^]).
 - Weitere Informationen finden Sie unter "[Bekannte Probleme](#)".

Schritt 1: Wählen Sie eine Version

Astra Trident Versionen folgen einem datumbasierten `YY.MM` Namensgebungskonvention, wobei „YY“ die letzten beiden Ziffern des Jahres und „MM“ der Monat ist. Dot Releases folgen einem `YY.MM.X` konvention, wo „X“ die Patch-Ebene ist. Sie wählen die Version, auf die Sie aktualisieren möchten, basierend auf der Version aus, von der Sie aktualisieren.

- Sie können ein direktes Upgrade auf jede Zielversion durchführen, die sich innerhalb eines Fensters mit vier Versionen Ihrer installierten Version befindet. Sie können beispielsweise direkt von 22.07 (oder einem beliebigen 22.07-Punkt-Release) auf 23.07 aktualisieren.
- Wenn Sie ein Upgrade von einer Version außerhalb des Fensters mit vier Releases durchführen, führen Sie ein Upgrade in mehreren Schritten durch. Befolgen Sie die Upgrade-Anweisungen für "[Frühere Version](#)" Sie führen ein Upgrade von auf die neueste Version durch, die für das Fenster mit vier Versionen geeignet ist. Wenn Sie beispielsweise 21.07 verwenden und ein Upgrade auf 23.07 durchführen möchten:

- a. Erstes Upgrade von 21.07 auf 22.07.
- b. Dann Upgrade von 22.07 auf 23.07.



Wenn Sie ein Upgrade über den Trident-Operator auf der OpenShift Container Platform durchführen, sollten Sie auf Trident 21.01.1 oder höher aktualisieren. Der mit 21.01.0 veröffentlichte Trident-Operator enthält ein bekanntes Problem, das in 21.01.1 behoben wurde. Weitere Informationen finden Sie im ["Details zur Ausgabe auf GitHub"](#).

Schritt 2: Bestimmen Sie die ursprüngliche Installationsmethode

So ermitteln Sie, welche Version Sie ursprünglich für Astra Trident verwendet haben:

1. Nutzung `kubectl get pods -n trident` Um die Pods zu untersuchen.
 - Wenn es keinen Operator Pod gibt, wurde Astra Trident mit `tridentctl` installiert.
 - Wenn es einen Operator Pod gibt, wurde Astra Trident entweder manuell oder über Helm mit dem Trident Operator installiert.
2. Wenn ein Benutzer-Pod vorhanden ist, verwenden Sie `kubectl describe tproc trident` Um festzustellen, ob Astra Trident mit Helm installiert wurde.
 - Wenn es ein Helm-Label gibt, wurde Astra Trident mit Helm installiert.
 - Wenn es kein Helm-Label gibt, wurde Astra Trident manuell über den Trident Operator installiert.

Schritt 3: Wählen Sie eine Upgrade-Methode

Im Allgemeinen sollten Sie das Upgrade mit der gleichen Methode durchführen, die Sie für die Erstinstallation verwendet haben, wie Sie es können ["Wechseln Sie zwischen den Installationsmethoden"](#). Astra Trident bietet zwei Optionen für ein Upgrade.

- ["Upgrade über den Trident-Operator"](#)



Wir empfehlen Ihnen, dies zu überprüfen ["Den Upgrade-Workflow für Bediener verstehen"](#) Vor der Aktualisierung mit dem Bediener.

*

Upgrade mit dem Bediener

Den Upgrade-Workflow für Bediener verstehen

Bevor Sie ein Upgrade von Astra Trident mit dem Trident-Operator durchführen, sollten Sie sich über die während des Upgrades auftretenden Hintergrundprozesse informieren. Dies umfasst Änderungen am Trident Controller, am Controller Pod und an Node-Pods sowie am Node-DemonSet, die Rolling-Updates ermöglichen.

Bearbeitung von Trident Upgrades für Betreiber

Einer der vielen ["Vorteile der Verwendung des Trident-Bediener"](#) Die Installation und das Upgrade von Astra Trident erfolgt automatisch für Astra Trident und Kubernetes-Objekte, ohne vorhandene gemountete Volumes zu unterbrechen. So kann Astra Trident Upgrades ohne Ausfallzeiten oder auch ohne ["Rollierende Updates"](#). Insbesondere kommuniziert der Trident Betreiber mit dem Kubernetes-Cluster, um:

- Löschen Sie die Trident Controller-Implementierung und den Node DemonSet und erstellen Sie sie neu.
- Ersetzen Sie den Trident Controller Pod und die Trident Node Pods durch neue Versionen.
 - Wenn ein Node nicht aktualisiert wird, verhindert dies nicht, dass die verbleibenden Nodes aktualisiert werden.
 - Nur Nodes mit einem laufenden Trident Node Pod können Volumes mounten.



Weitere Informationen zur Architektur von Astra Trident auf dem Kubernetes-Cluster finden Sie unter "[Die Architektur von Astra Trident](#)".

Arbeitsablauf für die Benutzeraktualisierung

Wenn Sie ein Upgrade mit dem Trident Operator initiieren:

1. Der **Trident-Operator**:
 - a. Erkennt die aktuell installierte Version von Astra Trident (Version n).
 - b. Aktualisiert alle Kubernetes-Objekte einschließlich CRDs, RBAC und Trident SVC.
 - c. Löscht die Trident Controller-Bereitstellung für Version n .
 - d. Erstellt die Trident-Controller-Bereitstellung für Version $n+1$.
2. **Kubernetes** erstellt Trident Controller Pod für $n+1$.
3. Der **Trident-Operator**:
 - a. Löscht das Trident Node DemonSet für n . Der Operator wartet nicht auf die Beendigung des Node-Pod.
 - b. Erstellt den Trident Node Demonset für $n+1$.
4. **Kubernetes** erstellt Trident Node Pods auf Nodes, auf denen Trident Node Pod n nicht ausgeführt wird. So wird sichergestellt, dass auf einem Node nie mehr als ein Trident Node Pod einer beliebigen Version vorhanden ist.

Upgrade einer Trident-Bedienerinstallation

Sie können ein Upgrade von Astra Trident mit dem Trident Operator entweder manuell oder mit Helm durchführen. Sie können ein Upgrade von einer Trident Benutzerinstallation auf eine andere Trident Benutzerinstallation durchführen oder von einem durchführenden `tridentctl` Installation auf eine Trident-Operatorversion. Prüfen "[Wählen Sie eine Aktualisierungsmethode aus](#)" Vor dem Upgrade einer Trident-Benutzerinstallation.

Aktualisieren einer manuellen Installation

Sie können von einer Installation eines Trident Operators mit Cluster-Umfang auf eine andere Installation eines Trident Operators mit Cluster-Umfang aktualisieren. Alle Astra Trident Versionen 21.01 und höher verwenden einen Operator mit Cluster-Umfang.



Für ein Upgrade von Astra Trident, das mit dem Namespace-Scoped Operator (Versionen 20.07 bis 20.10) installiert wurde, verwenden Sie die Upgrade-Anweisungen für "[Ihre installierte Version](#)" Von Astra Trident zu erhalten.

Über diese Aufgabe

Trident bietet eine Bundle-Datei, mit der Sie den Operator installieren und zugehörige Objekte für Ihre Kubernetes-Version erstellen können.

- Verwenden Sie für Cluster mit Kubernetes 1.24 oder früheren Versionen "[Bundle_pre_1_25.yaml](#)".
- Verwenden Sie für Cluster mit Kubernetes 1.25 oder höher "[Bundle_Post_1_25.yaml](#)".

Bevor Sie beginnen

Stellen Sie sicher, dass Sie ein Kubernetes-Cluster ausführen "[Eine unterstützte Kubernetes Version](#)".

Schritte

1. Überprüfen Sie die Astra Trident Version:

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

2. Löschen Sie den Trident-Operator, der zur Installation der aktuellen Astra Trident-Instanz verwendet wurde. Wenn Sie beispielsweise ein Upgrade von 23.04 durchführen, führen Sie den folgenden Befehl aus:

```
kubectl delete -f 23.04/trident-installer/deploy/<bundle.yaml> -n trident
```

3. Wenn Sie Ihre Erstinstallation mit angepasst haben `TridentOrchestrator` Attribute, können Sie die bearbeiten `TridentOrchestrator` Objekt zum Ändern der Installationsparameter. Dies kann auch Änderungen umfassen, die an der Angabe gespiegelter Trident- und CSI-Image-Register für den Offline-Modus vorgenommen wurden, Debug-Protokolle aktivieren oder Geheimnisse für die Bildausziehung angeben.
4. Installieren Sie Astra Trident mit der richtigen YAML-Bundle-Datei für Ihre Umgebung, wo `<bundle.yaml>` ist `bundle_pre_1_25.yaml` Oder `bundle_post_1_25.yaml` Basierend auf Ihrer Kubernetes-Version Wenn Sie beispielsweise Astra Trident 23.07 installieren, führen Sie den folgenden Befehl aus:

```
kubectl create -f 23.07.1/trident-installer/deploy/<bundle.yaml> -n trident
```

Aktualisieren einer Helm-Installation

Sie können ein Upgrade für eine Astra Trident Helm Installation durchführen.



Wenn Sie ein Kubernetes-Cluster von 1.24 auf 1.25 oder höher aktualisieren, auf das Astra Trident installiert ist, müssen Sie Werte.yaml aktualisieren `excludePodSecurityPolicy` Bis `true` Oder hinzufügen `--set excludePodSecurityPolicy=true` Bis zum `helm upgrade` Befehl bevor Sie ein Upgrade des Clusters durchführen können.

Schritte

1. Laden Sie die neueste Version von Astra Trident herunter.
2. Verwenden Sie die `helm upgrade` Befehl wo `trident-operator-23.07.1.tgz` Gibt die Version an,

auf die Sie ein Upgrade durchführen möchten.

```
helm upgrade <name> trident-operator-23.07.1.tgz
```

Wenn Sie während der Erstinstallation alle nicht standardmäßigen Optionen festlegen (z. B. Private, gespiegelte Registryes für Trident- und CSI-Images), verwenden Sie `--set` Um sicherzustellen, dass diese Optionen im Upgrade-Befehl enthalten sind, werden die Werte andernfalls auf die Standardeinstellung zurückgesetzt.



Um beispielsweise den Standardwert von `tridentDebug` zu ändern, Ausführen des folgenden Befehls:

```
helm upgrade <name> trident-operator-23.07.1-custom.tgz --set  
tridentDebug=true
```

3. Laufen `helm list` Um zu überprüfen, ob sowohl die Karten- als auch die App-Version aktualisiert wurden. Laufen `tridentctl logs` Um alle Debug-Nachrichten zu überprüfen.

Upgrade von einem `tridentctl` Installation zum Trident-Operator

Sie können ein Upgrade auf die neueste Version des Trident-Operators von durchführen `tridentctl` Installation: Die vorhandenen Back-Ends und VES stehen automatisch zur Verfügung.



Bevor Sie zwischen den Installationsmethoden wechseln, lesen Sie die Informationen "[Wechseln zwischen den Installationsmethoden](#)"

Schritte

1. Laden Sie die neueste Version von Astra Trident herunter.

```
# Download the release required [23.07.1]  
mkdir 23.07.1  
cd 23.07.1  
wget  
https://github.com/NetApp/trident/releases/download/v22.01.1/trident-  
installer-23.07.1.tar.gz  
tar -xf trident-installer-23.07.1.tar.gz  
cd trident-installer
```

2. Erstellen Sie die `tridentorchestrator` CRD aus dem Manifest.

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

3. Stellen Sie den Clusteroperator im selben Namespace bereit.

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

serviceaccount/trident-operator created
clusterrole.rbac.authorization.k8s.io/trident-operator created
clusterrolebinding.rbac.authorization.k8s.io/trident-operator created
deployment.apps/trident-operator created
podsecuritypolicy.policy/tridentoperatorpods created

#Examine the pods in the Trident namespace
```

| NAME | READY | STATUS | RESTARTS | AGE |
|-------------------------------------|-------|---------|----------|-------|
| trident-controller-79df798bdc-m79dc | 6/6 | Running | 0 | 150d |
| trident-node-linux-xrst8 | 2/2 | Running | 0 | 150d |
| trident-operator-5574dbbc68-nthjv | 1/1 | Running | 0 | 1m30s |

4. Erstellen Sie ein TridentOrchestrator CR für die Installation von Astra Trident.

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

kubectl create -f deploy/crds/tridentorchestrator_cr.yaml

#Examine the pods in the Trident namespace
```

| NAME | READY | STATUS | RESTARTS | AGE |
|-----------------------------------|-------|---------|----------|-------|
| trident-csi-79df798bdc-m79dc | 6/6 | Running | 0 | 1m |
| trident-csi-xrst8 | 2/2 | Running | 0 | 1m |
| trident-operator-5574dbbc68-nthjv | 1/1 | Running | 0 | 5m41s |

5. Bestätigen Sie, dass das Upgrade von Trident auf die beabsichtigte Version durchgeführt wurde.

```
kubectl describe torc trident | grep Message -A 3

Message:          Trident installed
Namespace:        trident
Status:           Installed
Version:          v23.07.1
```

Upgrade mit tridentctl

Sie können mithilfe von ganz einfach eine bestehende Astra Trident Installation aufrüsten `tridentctl`.

Über diese Aufgabe

Deinstallation und Neuinstallation von Astra Trident fungiert als Upgrade. Bei der Deinstallation von Trident werden die von der Astra Trident Implementierung verwendeten Persistent Volume Claim (PVC) und Persistent Volume (PV) nicht gelöscht. PVS, die bereits bereitgestellt wurden, bleiben verfügbar, während Astra Trident offline ist. Astra Trident stellt Volumes für alle PVCs bereit, die in der Zwischenzeit erstellt werden, sobald sie wieder online sind.

Bevor Sie beginnen

Prüfen "[Wählen Sie eine Aktualisierungsmethode aus](#)" Vor der Aktualisierung mit `tridentctl`.

Schritte

1. Führen Sie den Deinstallationsbefehl in aus `tridentctl` So entfernen Sie alle mit Astra Trident verbundenen Ressourcen mit Ausnahme der CRDs und zugehörigen Objekte.

```
./tridentctl uninstall -n <namespace>
```

2. Installieren Sie Astra Trident Neu. Siehe "[Installieren Sie Astra Trident mit tridentctl](#)".



Unterbrechen Sie den Upgrade-Prozess nicht. Stellen Sie sicher, dass das Installationsprogramm bis zum Abschluss ausgeführt wird.

Überwachen Sie Astra Trident

Astra Trident bietet eine Reihe von Prometheus Kennzahlen-Endpunkten, mit denen Sie die Performance von Astra Trident überwachen können.

Überblick

Mit den von Astra Trident bereitgestellten Metriken können Sie:

- Bleiben Sie auf dem Laufenden über den Zustand und die Konfiguration von Astra Trident. Sie können prüfen, wie erfolgreich Vorgänge sind und ob sie wie erwartet mit den Back-Ends kommunizieren können.
- Untersuchen Sie die Back-End-Nutzungsinformationen und erfahren Sie, wie viele Volumes auf einem Back-End bereitgestellt werden, sowie den belegten Speicherplatz usw.
- Erstellt eine Zuordnung der Anzahl von Volumes, die über verfügbare Back-Ends bereitgestellt werden.
- Verfolgen Sie die Leistung. Sie können sich ansehen, wie lange Astra Trident für die Kommunikation mit Back-Ends und die Durchführung von Vorgängen benötigt.



Die Metriken von Trident sind standardmäßig auf dem Ziel-Port offengelegt 8001 Am `/metrics` endpunkt: Diese Metriken sind bei der Installation von Trident standardmäßig aktiviert.

Was Sie benötigen

- Kubernetes-Cluster mit installiertem Astra Trident
- Eine Prometheus Instanz. Dies kann ein sein ["Implementierung von Container-Prometheus"](#) Oder Sie können Prometheus als ein ausführen ["Native Applikation"](#).

Schritt 1: Definieren Sie ein Prometheus-Ziel

Sie sollten ein Prometheus Ziel definieren, um die Kennzahlen zu sammeln und Informationen über das Management von Back-Ends Astra Trident, die von ihm erstellten Volumes usw. zu erhalten. Das ["Blog"](#) Erläutert, wie Sie mithilfe von Prometheus und Grafana mit Astra Trident Kennzahlen abrufen können. Im Blog erfahren Sie, wie Sie Prometheus als Betreiber in Ihrem Kubernetes-Cluster ausführen und einen ServiceMonitor erstellen können, um Astra Trident-Kennzahlen zu erhalten.

Schritt: Erstellen Sie einen Prometheus ServiceMonitor

Um die Trident Kennzahlen zu verwenden, sollten Sie ein Prometheus ServiceMonitor erstellen, das überwacht `trident-csi` Service und wartet auf den `metrics` Port: Ein Beispiel für ServiceMonitor sieht so aus:

```
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  name: trident-sm
  namespace: monitoring
  labels:
    release: prom-operator
spec:
  jobLabel: trident
  selector:
    matchLabels:
      app: controller.csi.trident.netapp.io
  namespaceSelector:
    matchNames:
      - trident
  endpoints:
    - port: metrics
      interval: 15s
```

Diese ServiceMonitor-Definition ruft vom zurückgegebene Kennzahlen ab `trident-csi` Service und insbesondere sucht nach dem `metrics` endpoint des Dienstes: Daher ist Prometheus jetzt so konfiguriert, dass es die von Astra Trident verstehen wird Kennzahlen.

Neben den direkt bei Astra Trident verfügbaren Kennzahlen gibt kubelet auch viele andere Lösungen auf `kubelet_volume_*` Kennzahlen über den Endpunkt der IT-eigenen Kennzahlen. Kubelet kann Informationen über verbundene Volumes bereitstellen und Pods und andere interne Vorgänge, die er übernimmt. Siehe ["Hier"](#).

Schritt 3: Abfrage der Trident-Kennzahlen mit PromQL

PromQL ist gut geeignet, um Ausdrücke zu erstellen, die Zeitreihen- oder tabellarische Daten zurückgeben.

Im Folgenden finden Sie einige PromQL-Abfragen, die Sie verwenden können:

Abrufen des Integritätsinformationen zu Trident

- **Prozentsatz der HTTP 2XX-Antworten von Astra Trident**

```
(sum (trident_rest_ops_seconds_total_count{status_code=~"2.."} OR on()  
vector(0)) / sum (trident_rest_ops_seconds_total_count)) * 100
```

- **Prozentualer Anteil DER REST-Antworten von Astra Trident über Statuscode**

```
(sum (trident_rest_ops_seconds_total_count) by (status_code) / scalar  
(sum (trident_rest_ops_seconds_total_count))) * 100
```

- **Durchschnittsdauer in ms der von Astra Trident durchgeführten Operationen**

```
sum by (operation)  
(trident_operation_duration_milliseconds_sum{success="true"}) / sum by  
(operation)  
(trident_operation_duration_milliseconds_count{success="true"})
```

Holen Sie sich Informationen zur Nutzung von Astra Trident

- **Mittlere Volumengröße**

```
trident_volume_allocated_bytes/trident_volume_count
```

- **Gesamter Volume-Speicherplatz, der von jedem Backend bereitgestellt wird**

```
sum (trident_volume_allocated_bytes) by (backend_uuid)
```

Individuelle Volume-Nutzung



Dies ist nur aktiviert, wenn auch kubelet-Kennzahlen gesammelt werden.

- **Prozentsatz des verwendeten Speicherplatzes für jedes Volumen**

```
kubelet_volume_stats_used_bytes / kubelet_volume_stats_capacity_bytes *
100
```

AutoSupport Telemetrie von Astra Trident mit Thema

Standardmäßig sendet Astra Trident in einem täglichen Intervall Prometheus-Kennzahlen und grundlegende Backend-Informationen an NetApp.

- Um zu verhindern, dass Astra Trident die Prometheus Kennzahlen und grundlegende Backend-Informationen an NetApp sendet, bestehen Sie am `--silence-autosupport` Fahne während der Installation von Astra Trident.
- Astra Trident kann auch Container-Protokolle per On-Demand an den NetApp Support senden `tridentctl send autosupport`. Sie müssen Astra Trident auslösen, um seine Protokolle hochzuladen. Bevor Sie Protokolle einreichen, sollten Sie die von NetApp akzeptieren ["datenschutzrichtlinie"](#).
- Sofern nicht angegeben, ruft Astra Trident die Protokolle der letzten 24 Stunden ab.
- Sie können den Zeitrahmen für die Protokollaufbewahrung mit festlegen `--since` Flagge. Beispiel: `tridentctl send autosupport --since=1h`. Diese Informationen werden über ein gesammelt und versendet `trident-autosupport` Container
Das wird neben Astra Trident installiert. Sie können das Container-Image unter abrufen ["Trident AutoSupport"](#).
- Trident AutoSupport erfasst oder übermittelt keine personenbezogenen Daten oder personenbezogenen Daten. Es kommt mit einem ["EULA"](#) Dies gilt nicht für das Trident Container-Image selbst. Weitere Informationen zum Engagement von NetApp für Datensicherheit und Vertrauen ["Hier"](#).

Eine von Astra Trident gesendete Beispiellast sieht folgendermaßen aus:

```
---
items:
- backendUUID: ff3852e1-18a5-4df4-b2d3-f59f829627ed
  protocol: file
  config:
    version: 1
    storageDriverName: ontap-nas
    debug: false
    debugTraceFlags:
    disableDelete: false
    serialNumbers:
    - nwkvzfanek_SN
    limitVolumeSize: ''
  state: online
  online: true
```

- Die AutoSupport Meldungen werden an den AutoSupport Endpunkt von NetApp gesendet. Wenn Sie zum Speichern von Container-Images eine private Registrierung verwenden, können Sie das verwenden `--image-registry` Flagge.

- Sie können auch Proxy-URLs konfigurieren, indem Sie die Installation YAML-Dateien erstellen. Dies kann mit `tridentctl install --generate-custom-yaml` So erstellen Sie die YAML-Dateien und fügen die hinzu `--proxy-url` Argument für das `trident-autosupport` Container in `trident-deployment.yaml`.

Deaktivieren Sie Astra Trident Metriken

Um `**`-Metriken von der Meldung zu deaktivieren, sollten Sie benutzerdefinierte YAML generieren (mit dem `--generate-custom-yaml` Markieren) und bearbeiten, um die zu entfernen `--metrics` Flagge wird für das aufgerufen `trident-main` Container:

Deinstallieren Sie Astra Trident

Sie sollten die gleiche Methode verwenden, um Astra Trident zu deinstallieren, die Sie zur Installation von Astra Trident verwendet haben.

Über diese Aufgabe

- Wenn Sie eine Fehlerbehebung für Fehler benötigen, die nach einem Upgrade, Abhängigkeitsproblemen oder einem nicht erfolgreichen oder unvollständigen Upgrade beobachtet wurden, sollten Sie Astra Trident deinstallieren und die frühere Version mithilfe der entsprechenden Anweisungen neu installieren "[Version](#)". Dies ist die einzige empfohlene Möglichkeit, *Downgrade* auf eine frühere Version zu übertragen.
- Für eine einfache Aktualisierung und Neuinstallation entfernt das Deinstallieren von Astra Trident nicht die CRDs oder damit verbundene Objekte, die von Astra Trident erstellt wurden. Wenn Sie Astra Trident und alle zugehörigen Daten vollständig entfernen müssen, finden Sie weitere Informationen unter "[Entfernen Sie Astra Trident und CRDs vollständig](#)".

Bevor Sie beginnen

Falls Sie Kubernetes-Cluster stilllegen, müssen Sie alle Applikationen löschen, die Volumes verwenden, die von Astra Trident erstellt wurden, bevor Sie sie deinstallieren. Dadurch wird sichergestellt, dass PVCs auf Kubernetes-Nodes nicht veröffentlicht werden, bevor sie gelöscht werden.

Bestimmen Sie die ursprüngliche Installationsmethode

Sie sollten die gleiche Methode verwenden, um Astra Trident zu deinstallieren, die Sie verwendet haben, um es zu installieren. Überprüfen Sie vor der Deinstallation, mit welcher Version Sie Astra Trident ursprünglich installiert haben.

1. Nutzung `kubectl get pods -n trident` Um die Pods zu untersuchen.
 - Wenn es keinen Operator Pod gibt, wurde Astra Trident mit installiert `tridentctl`.
 - Wenn es einen Operator Pod gibt, wurde Astra Trident entweder manuell oder über Helm mit dem Trident Operator installiert.
2. Wenn ein Benutzer-Pod vorhanden ist, verwenden Sie `kubectl describe tproc trident` Um festzustellen, ob Astra Trident mit Helm installiert wurde.
 - Wenn es ein Helm-Label gibt, wurde Astra Trident mit Helm installiert.
 - Wenn es kein Helm-Label gibt, wurde Astra Trident manuell über den Trident Operator installiert.

Deinstallieren Sie die Installation eines Trident-Operators

Sie können die Installation eines Dreizack-Bedieners manuell oder mithilfe von Helm deinstallieren.

Deinstallieren Sie die manuelle Installation

Falls Sie Astra Trident mit dem Operator installiert haben, können Sie es deinstallieren, indem Sie einen der folgenden Schritte ausführen:

1. Bearbeiten `TridentOrchestrator` CR und stellen Sie das Deinstallationsflag ein:

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

Wenn der `uninstall` Flag ist auf festgelegt `true`, Der Trident-Operator deinstalliert Trident, entfernt jedoch nicht den `tridentOrchestrator` selbst. Sie sollten den `TridentOrchestrator` aufräumen und einen neuen erstellen, wenn Sie Trident erneut installieren möchten.

- Löschen `TridentOrchestrator`:** Durch Entfernen der `TridentOrchestrator` CR, das zur Implementierung von Astra Trident verwendet wurde, weisen Sie den Bediener an, Trident zu deinstallieren. Der Bediener verarbeitet die Entfernung von `TridentOrchestrator` Außerdem wird die Implementierung und das Dämonenset Astra Trident entfernt und die im Rahmen der Installation erstellten Trident-Pods gelöscht.

```
kubectl delete -f deploy/<bundle.yaml> -n <namespace>
```

Deinstallieren Sie Helm-Installation

Wenn Sie Astra Trident mithilfe von Helm installiert haben, können Sie es mit `deinstallieren helm uninstall`.

```
#List the Helm release corresponding to the Astra Trident install.
helm ls -n trident
NAME                NAMESPACE      REVISION      UPDATED
STATUS              CHART           APP VERSION
trident             trident        1             2021-04-20
00:26:42.417764794 +0000 UTC deployed      trident-operator-21.07.1
21.07.1

#Uninstall Helm release to remove Trident
helm uninstall trident -n trident
release "trident" uninstalled
```

Deinstallieren Sie A `tridentctl` Installation

Verwenden Sie die `uninstall` Befehl in `tridentctl` So entfernen Sie alle Ressourcen, die Astra Trident mit

Ausnahme der CRDs und zugehörigen Objekte zugeordnet sind:

```
./tridentctl uninstall -n <namespace>
```

Astra Trident für Docker

Voraussetzungen für die Bereitstellung

Bevor Sie Astra Trident implementieren können, müssen Sie die erforderlichen Protokollvoraussetzungen auf Ihrem Host installieren und konfigurieren.


Überprüfen Sie die Anforderungen

- Stellen Sie sicher, dass Ihre Implementierung alle Anforderungen erfüllt "[Anforderungen](#)".
- Vergewissern Sie sich, dass eine unterstützte Version von Docker installiert ist. Wenn Ihre Docker Version veraltet ist, "[Installieren oder aktualisieren Sie sie](#)".

```
docker --version
```

- Vergewissern Sie sich, dass die Protokollvoraussetzungen auf Ihrem Host installiert und konfiguriert sind:

| Protokoll | Betriebssystem | Befehle |
|-----------|-------------------|---|
| NFS | RHEL 8 ODER HÖHER | <code>sudo yum install -y nfs-utils</code> |
| NFS | Ubuntu | <code>sudo apt-get install -y nfs-common</code> |

| Protokoll | Betriebssystem | Befehle |
|-----------|-------------------|---|
| ISCSI | RHEL 8 ODER HÖHER | <p>1. Installieren Sie die folgenden Systempakete:</p> <pre>sudo yum install -y lsscsi iscsi-initiator- utils sg3_utils device- mapper-multipath</pre> <p>2. Überprüfen Sie, ob die Version von iscsi-Initiator-utils 6.2.0.874-2.el7 oder höher ist:</p> <pre>rpm -q iscsi-initiator- utils</pre> <p>3. Scannen auf manuell einstellen:</p> <pre>sudo sed -i 's/^\(node.session.scan \).*\/\1 = manual/' /etc/iscsi/iscsid.conf</pre> <p>4. Multipathing aktivieren:</p> <pre>sudo mpathconf --enable --with_multipathd y --find_multipaths n</pre> <div style="border: 1px solid gray; padding: 5px; margin: 10px 0;">  <p>Unbedingt etc/multipat h.conf Enthält find_multipa ths no Unter defaults.</p> </div> <p>5. Stellen Sie das sicher iscsid Und multipathd Laufen:</p> <pre>sudo systemctl enable --now iscsid multipathd</pre> <p>6. Aktivieren und starten iscsi:</p> <pre>sudo systemctl enable --now iscsi</pre> |

| Protokoll | Betriebssystem | Befehle |
|-----------|----------------|---|
| ISCSI | Ubuntu | <p>1. Installieren Sie die folgenden Systempakete:</p> <pre>sudo apt-get install -y open-iscsi lsscsi sg3-utils multipath-tools scsitools</pre> <p>2. Stellen Sie sicher, dass Open-iscsi-Version 2.0.874-5ubuntu2.10 oder höher (für bionic) oder 2.0.874-7.1ubuntu6.1 oder höher (für Brennweite) ist:</p> <pre>dpkg -l open-iscsi</pre> <p>3. Scannen auf manuell einstellen:</p> <pre>sudo sed -i 's/^\(node.session.scan\) .*/\1 = manual/' /etc/iscsi/iscsid.conf</pre> <p>4. Multipathing aktivieren:</p> <pre>sudo tee /etc/multipath.conf < ←'EOF' defaults { user_friendly_names yes find_multipaths no } EOF sudo systemctl enable --now multipath-tools.service sudo service multipath-tools restart</pre> <div style="border: 1px solid gray; padding: 5px; margin: 10px 0;"> <p> Unbedingt etc/multipath.conf Enthält find_multipaths no Unter defaults.</p> </div> <p>5. Stellen Sie das sicher open-iscsi Und multipath-tools Sind aktiviert und läuft:</p> <pre>sudo systemctl status multipath-tools</pre> |

Implementieren Sie Astra Trident

Astra Trident für Docker bietet eine direkte Integration in das Docker Ecosystem für NetApp Storage-Plattformen. Die Plattform unterstützt auch das Provisioning und Management von Storage-Ressourcen – von der Storage-Plattform bis hin zu Docker Hosts – mit einem Framework für zukünftige zusätzliche Plattformen.

Mehrere Instanzen von Astra Trident können gleichzeitig auf demselben Host ausgeführt werden. Dies ermöglicht simultane Verbindungen zu mehreren Storage-Systemen und Storage-Typen und kann den für die Docker Volumes verwendeten Storage angepasst werden.

Was Sie benötigen

Siehe "[Voraussetzungen für die Bereitstellung](#)". Wenn Sie die Voraussetzungen erfüllt haben, können Sie Astra Trident implementieren.

Docker Managed Plug-in-Methode (Version 1.13/17.03 und höher)

Bevor Sie beginnen



Wenn Sie Astra Trident vor Docker 1.13/17.03 in der herkömmlichen Daemon-Methode verwendet haben, stellen Sie sicher, dass Sie den Astra Trident-Prozess beenden und Ihren Docker-Daemon neu starten, bevor Sie die Managed Plug-in-Methode verwenden.

1. Beenden Sie alle laufenden Instanzen:

```
killall /usr/local/bin/netappdvp  
killall /usr/local/bin/trident
```

2. Docker Neu Starten.

```
systemctl restart docker
```

3. Vergewissern Sie sich, dass Docker Engine 17.03 (neu 1.13) oder höher installiert ist.

```
docker --version
```

Wenn Ihre Version veraltet ist, "[Installieren oder aktualisieren Sie Ihre Installation](#)".

Schritte

1. Erstellen Sie eine Konfigurationsdatei und geben Sie die Optionen wie folgt an:

- `config`: Der Standarddateiname ist `config.json`, Sie können jedoch einen beliebigen Namen verwenden, den Sie wählen, indem Sie die `config` Option mit dem Dateinamen. Die Konfigurationsdatei muss im enthalten sein `/etc/netappdvp` Verzeichnis auf dem Hostsystem.
- `log-level`: Geben Sie die Protokollierungsebene an (`debug`, `info`, `warn`, `error`, `fatal`). Die Standardeinstellung lautet `info`.

- debug: Geben Sie an, ob Debug-Protokollierung aktiviert ist. Die Standardeinstellung lautet false. Überschreibt die Protokollebene, wenn wahr.

- i. Speicherort für die Konfigurationsdatei erstellen:

```
sudo mkdir -p /etc/netappdvp
```

- ii. Konfigurationsdatei erstellen:

```
cat << EOF > /etc/netappdvp/config.json
{
  "version": 1,
  "storageDriverName": "ontap-nas",
  "managementLIF": "10.0.0.1",
  "dataLIF": "10.0.0.2",
  "svm": "svm_nfs",
  "username": "vsadmin",
  "password": "password",
  "aggregate": "aggr1"
}
EOF
```

2. Starten Sie Astra Trident mit dem Managed Plug-in-System. Austausch `<version>` Mit der Plugin-Version (xxx.xx.x) verwenden Sie.

```
docker plugin install --grant-all-permissions --alias netapp
netapp/trident-plugin:<version> config=myConfigFile.json
```

3. Beginnen Sie mit Astra Trident, um Storage aus dem konfigurierten System zu nutzen.

- a. Erstellen Sie ein Volume mit dem Namen „FirstVolume“:

```
docker volume create -d netapp --name firstVolume
```

- b. Erstellen Sie ein Standardvolume beim Starten des Containers:

```
docker run --rm -it --volume-driver netapp --volume
secondVolume:/my_vol alpine ash
```

- c. Entfernen Sie den Datenträger „FirstVolume“:

```
docker volume rm firstVolume
```

Herkömmliche Methode (Version 1.12 oder früher)

Bevor Sie beginnen

1. Stellen Sie sicher, dass Sie Docker Version 1.10 oder höher haben.

```
docker --version
```

Wenn Ihre Version veraltet ist, aktualisieren Sie Ihre Installation.

```
curl -fsSL https://get.docker.com/ | sh
```

Oder "[Befolgen Sie die Anweisungen für Ihre Distribution](#)".

2. Stellen Sie sicher, dass NFS und/oder iSCSI für Ihr System konfiguriert ist.

Schritte

1. NetApp Docker Volume Plug-in installieren und konfigurieren:
 - a. Laden Sie die Anwendung herunter und entpacken Sie sie:

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

- b. Verschieben Sie zu einer Position im bin-Pfad:

```
sudo mv trident-installer/extras/bin/trident /usr/local/bin/  
sudo chown root:root /usr/local/bin/trident  
sudo chmod 755 /usr/local/bin/trident
```

- c. Speicherort für die Konfigurationsdatei erstellen:

```
sudo mkdir -p /etc/netappdvp
```

- d. Konfigurationsdatei erstellen:

```
cat << EOF > /etc/netappdvp/ontap-nas.json
{
  "version": 1,
  "storageDriverName": "ontap-nas",
  "managementLIF": "10.0.0.1",
  "dataLIF": "10.0.0.2",
  "svm": "svm_nfs",
  "username": "vsadmin",
  "password": "password",
  "aggregate": "aggr1"
}
EOF
```

2. Nachdem Sie die Binärdatei platziert und die Konfigurationsdatei erstellt haben, starten Sie den Trident-Daemon mit der gewünschten Konfigurationsdatei.

```
sudo trident --config=/etc/netappdvp/ontap-nas.json
```



Sofern nicht angegeben, lautet der Standardname für den Volume-Treiber „NetApp“.

Nachdem der Daemon gestartet wurde, können Sie Volumes mithilfe der Docker CLI-Schnittstelle erstellen und verwalten

3. Volume erstellen:

```
docker volume create -d netapp --name trident_1
```

4. Bereitstellung eines Docker Volumes beim Starten eines Containers:

```
docker run --rm -it --volume-driver netapp --volume trident_2:/my_vol
alpine ash
```

5. Entfernen eines Docker Volumes:

```
docker volume rm trident_1
docker volume rm trident_2
```

Starten Sie Astra Trident beim Systemstart

Eine Beispieldatei für systembasierte Systeme finden Sie unter `contrib/trident.service.example` im Git Repo. Gehen Sie wie folgt vor, um die Datei mit RHEL zu verwenden:

1. Kopieren Sie die Datei an den richtigen Speicherort.

Sie sollten eindeutige Namen für die Einheitendateien verwenden, wenn mehr als eine Instanz ausgeführt wird.

```
cp contrib/trident.service.example
/usr/lib/systemd/system/trident.service
```

2. Bearbeiten Sie die Datei, ändern Sie die Beschreibung (Zeile 2) entsprechend dem Treibernamen und dem Konfigurationspfad (Zeile 9), um Ihre Umgebung zu berücksichtigen.
3. Systemd neu laden, damit sie Änderungen aufnehmen kann:

```
systemctl daemon-reload
```

4. Aktivieren Sie den Service.

Dieser Name variiert je nach Namen der Datei in `/usr/lib/systemd/system` Verzeichnis.

```
systemctl enable trident
```

5. Starten Sie den Service.

```
systemctl start trident
```

6. Den -Status anzeigen.

```
systemctl status trident
```



Wenn Sie die Einheitendatei ändern, führen Sie den aus `systemctl daemon-reload` Befehl, damit sie die Änderungen kennt.

Astra Trident upgraden oder deinstallieren

Astra Trident ist ohne Auswirkungen auf die verwendeten Volumes sicher auf Docker aktualisieren zu können. Während des Upgrade-Prozesses gibt es eine kurze Zeit, wo `docker volume` Befehle, die an das Plugin gerichtet werden, werden nicht erfolgreich sein, und Anwendungen werden nicht in der Lage sein, Volumes zu mounten, bis das Plugin wieder ausgeführt wird. Unter den meisten Umständen dauert das nur wenige Sekunden.

Upgrade

Führen Sie die nachstehenden Schritte zum Upgrade von Astra Trident für Docker durch.

Schritte

1. Liste der vorhandenen Volumes:

```
docker volume ls
DRIVER          VOLUME NAME
netapp:latest   my_volume
```

2. Deaktivieren Sie das Plugin:

```
docker plugin disable -f netapp:latest
docker plugin ls
ID              NAME          DESCRIPTION
ENABLED
7067f39a5df5   netapp:latest nDVP - NetApp Docker Volume
Plugin        false
```

3. Upgrade des Plug-ins:

```
docker plugin upgrade --skip-remote-check --grant-all-permissions
netapp:latest netapp/trident-plugin:21.07
```



Die Version 18.01 von Astra Trident ersetzt die nDVP. Sie sollten ein Upgrade direkt von durchführen `netapp/ndvp-plugin` Bild an den `netapp/trident-plugin` Bild:

4. Plug-in aktivieren:

```
docker plugin enable netapp:latest
```

5. Vergewissern Sie sich, dass das Plug-in aktiviert ist:

```
docker plugin ls
ID              NAME          DESCRIPTION
ENABLED
7067f39a5df5   netapp:latest Trident - NetApp Docker Volume
Plugin        true
```

6. Vergewissern Sie sich, dass die Volumes sichtbar sind:

```
docker volume ls
DRIVER          VOLUME NAME
netapp:latest   my_volume
```



Wenn Sie ein Upgrade von einer alten Version von Astra Trident (vor 20.10) auf Astra Trident 20.10 oder höher durchführen, tritt möglicherweise ein Fehler auf. Weitere Informationen finden Sie unter "[Bekannte Probleme](#)". Wenn der Fehler auftritt, sollten Sie zuerst das Plugin deaktivieren, dann das Plugin entfernen und dann die erforderliche Astra Trident Version installieren, indem Sie einen zusätzlichen Konfigurationsparameter übergeben: `docker plugin install netapp/trident-plugin:20.10 --alias netapp --grant-all -permissions config=config.json`

Deinstallieren

Führen Sie die folgenden Schritte aus, um Astra Trident für Docker zu deinstallieren.

Schritte

1. Entfernen Sie alle Volumes, die das Plugin erstellt.
2. Deaktivieren Sie das Plugin:

```
docker plugin disable netapp:latest
docker plugin ls
ID                NAME                DESCRIPTION
ENABLED
7067f39a5df5     netapp:latest      nDVP - NetApp Docker Volume
Plugin   false
```

3. Entfernen Sie das Plugin:

```
docker plugin rm netapp:latest
```

Arbeiten mit Volumes

Volumes lassen sich ganz einfach mit dem Standard erstellen, klonen und entfernen `docker volume` Befehle mit dem bei Bedarf angegebenen Astra Trident-Treibernamen.

Erstellen eines Volumes

- Erstellen Sie ein Volume mit einem Treiber unter Verwendung des Standardnamens:

```
docker volume create -d netapp --name firstVolume
```


- Erstellung eines Volumes mit einer bestimmten Astra Trident Instanz:

```
docker volume create -d ntap_bronze --name bronzeVolume
```



Falls Sie keine angeben "**Optionen**", Die Standardeinstellungen für den Treiber werden verwendet.

- Überschreiben Sie die Standard-Volume-Größe. Beachten Sie das folgende Beispiel, um ein 20 gib-Volume mit einem Treiber zu erstellen:

```
docker volume create -d netapp --name my_vol --opt size=20G
```



Die Volume-Größen werden als Strings angegeben, die einen ganzzahligen Wert mit optionalen Einheiten enthalten (Beispiel: 10G, 20GB, 3tib). Wenn keine Einheiten angegeben werden, lautet der Standardwert G. Einheiten der Größe können entweder als Befugnisse von 2 (B, KiB, MiB, gib, tib) oder als Befugnis von 10 (B, KB, MB, GB, TB) angegeben werden. Auf Kurzschluss und Einheiten werden 2 Kräfte (G = gib, T = tib, ...) verwendet.

Entfernen Sie ein Volume

- Entfernen Sie das Volume wie jedes andere Docker Volume:

```
docker volume rm firstVolume
```



Bei Verwendung des `solidfire-san` Treiber, im obigen Beispiel wird das Volume gelöscht und gelöscht.

Führen Sie die nachstehenden Schritte zum Upgrade von Astra Trident für Docker durch.

Klonen Sie ein Volume

Bei Verwendung des `ontap-nas`, `ontap-san`, `solidfire-san`, und `gcp-cvs storage drivers`Astra Trident` kann Volumes klonen. Bei Verwendung des `ontap-nas-flexgroup` Oder `ontap-nas-economy` Treiber, Klonen wird nicht unterstützt. Wenn Sie ein neues Volume von einem vorhandenen Volume erstellen, wird ein neuer Snapshot erstellt.

- Überprüfen Sie das Volume, um die Snapshots aufzuzählen:

```
docker volume inspect <volume_name>
```

- Erstellen Sie ein neues Volume von einem vorhandenen Volume aus. Dadurch wird ein neuer Snapshot erstellt:

```
docker volume create -d <driver_name> --name <new_name> -o
from=<source_docker_volume>
```

- Erstellen Sie ein neues Volume anhand eines vorhandenen Snapshots auf einem Volume. Dadurch wird kein neuer Snapshot erstellt:

```
docker volume create -d <driver_name> --name <new_name> -o
from=<source_docker_volume> -o fromSnapshot=<source_snap_name>
```

Beispiel

```
docker volume inspect firstVolume
```

```
[
  {
    "Driver": "ontap-nas",
    "Labels": null,
    "Mountpoint": "/var/lib/docker-volumes/ontap-
nas/netappdvp_firstVolume",
    "Name": "firstVolume",
    "Options": {},
    "Scope": "global",
    "Status": {
      "Snapshots": [
        {
          "Created": "2017-02-10T19:05:00Z",
          "Name": "hourly.2017-02-10_1505"
        }
      ]
    }
  }
]
```

```
docker volume create -d ontap-nas --name clonedVolume -o from=firstVolume
clonedVolume
```

```
docker volume rm clonedVolume
```

```
docker volume create -d ontap-nas --name volFromSnap -o from=firstVolume
-o fromSnapshot=hourly.2017-02-10_1505
volFromSnap
```

```
docker volume rm volFromSnap
```

Zugriff auf extern erstellte Volumes

Mit Trident können Sie auf extern erstellte Blockgeräte (oder deren Klone) von Containern zugreifen, die Trident verwenden **nur**, wenn sie keine Partitionen haben und ihr Dateisystem von Astra Trident unterstützt wird (z.B. an ext4-Formatiert /dev/sdc1 Nicht über Astra Trident zugänglich).

Treiberspezifische Volume-Optionen

Jeder Storage-Treiber verfügt über unterschiedliche Optionen, die Sie bei der Volume-Erstellung angeben können, um das Ergebnis anzupassen. Unter finden Sie weitere Optionen, die für Ihr konfiguriertes Storage-System gelten.

Die Verwendung dieser Optionen während der Erstellung des Volumes ist einfach. Geben Sie die Option und den Wert über das `-o` Operator während des CLI-Vorgangs. Diese überschreiben alle gleichwertigen Werte aus der JSON-Konfigurationsdatei.

ONTAP Volume-Optionen

Bei der Erstellung von Volumes für NFS und iSCSI sind folgende Optionen enthalten:

| Option | Beschreibung |
|-----------------------------|---|
| <code>size</code> | Die Größe des Volumes beträgt standardmäßig 1 gib. |
| <code>spaceReserve</code> | Thin oder Thick Provisioning stellen das Volume bereit. Die Standardeinstellung ist „Thin“. Gültige Werte sind <code>none</code> (Thin Provisioning) und <code>volume</code> (Thick Provisioning): |
| <code>snapshotPolicy</code> | Dadurch wird die Snapshot-Richtlinie auf den gewünschten Wert eingestellt. Die Standardeinstellung lautet <code>none</code> , Das bedeutet, dass keine Snapshots automatisch für den Datenträger erstellt werden. Sofern nicht von Ihrem Speicheradministrator geändert, existiert eine Richtlinie namens „Standard“ auf allen ONTAP Systemen, die sechs stündliche, zwei tägliche und zwei wöchentliche Schnappschüsse erzeugt und speichert. Die in einem Snapshot erhaltenen Daten können durch Durchsuchen der wiederhergestellt werden <code>.snapshot</code> Verzeichnis in einem beliebigen Verzeichnis im Volume. |

| Option | Beschreibung |
|-----------------|--|
| snapshotReserve | Dadurch wird die Snapshot-Reserve auf den gewünschten Prozentsatz eingestellt. Der Standardwert ist kein Wert, was bedeutet, dass ONTAP die Snapshot Reserve (in der Regel 5%) auswählen wird, wenn Sie eine Snapshot Policy ausgewählt haben, oder 0%, wenn die Snapshot Policy keine ist. Sie können den Standardwert von snapshotReserve in der Konfigurationsdatei für alle ONTAP-Back-Ends setzen und es als Option zur Erstellung von Volumes für alle ONTAP-Back-Ends außer ontap-nas-Economy verwenden. |
| splitOnClone | Beim Klonen eines Volume wird dadurch ONTAP den Klon sofort von seinem übergeordneten Volume aufteilen. Die Standardeinstellung lautet <code>false</code> . Einige Anwendungsfälle für das Klonen von Volumes werden am besten bedient, indem der Klon unmittelbar nach der Erstellung von seinem übergeordneten Volume aufgeteilt wird, da sich die Storage-Effizienz wahrscheinlich nicht erhöhen wird. Das Klonen einer leeren Datenbank spart beispielsweise viel Zeit, spart aber nur wenig Storage. Daher ist es am besten, den Klon sofort zu teilen. |
| encryption | <p>Aktivieren Sie NetApp Volume Encryption (NVE) auf dem neuen Volume, standardmäßig aktiviert <code>false</code>. NVE muss im Cluster lizenziert und aktiviert sein, damit diese Option verwendet werden kann.</p> <p>Wenn NAE auf dem Backend aktiviert ist, wird jedes im Astra Trident bereitgestellte Volume NAE aktiviert.</p> <p>Weitere Informationen finden Sie unter: "Astra Trident arbeitet mit NVE und NAE zusammen".</p> |
| tieringPolicy | Legt die Tiering-Richtlinie fest, die für das Volume verwendet werden soll. Damit wird entschieden, ob Daten auf die Cloud-Tier verschoben werden, wenn sie inaktiv sind (kalte). |

Die folgenden zusätzlichen Optionen sind nur für NFS* verfügbar:

| Option | Beschreibung |
|------------------------------|---|
| <code>unixPermissions</code> | Dadurch wird der Berechtigungssatz für das Volume selbst festgelegt. Standardmäßig werden die Berechtigungen auf festgelegt <code>---rwxr-xr-x</code> , Oder in numerischer Notation <code>0755</code> , und <code>root</code> Wird der Eigentümer sein. Das Text- oder Zahlenformat funktioniert. |
| <code>snapshotDir</code> | Einstellen auf <code>true</code> Wird die schaffen <code>.snapshot</code> Für Clients, die auf das Volume zugreifen, sichtbares Verzeichnis Der Standardwert ist <code>false</code> , Das bedeutet, dass die Sichtbarkeit des <code>.snapshot</code> Das Verzeichnis ist standardmäßig deaktiviert. Einige Bilder, zum Beispiel das offizielle MySQL-Image, funktionieren nicht wie erwartet, wenn das <code>.snapshot</code> Verzeichnis wird angezeigt. |
| <code>exportPolicy</code> | Legt die Exportrichtlinie fest, die für das Volume verwendet werden soll. Die Standardeinstellung lautet <code>default</code> . |
| <code>securityStyle</code> | Legt den Sicherheitsstil für den Zugriff auf das Volume fest. Die Standardeinstellung lautet <code>unix</code> . Gültige Werte sind <code>unix</code> Und <code>mixed</code> . |

Die folgenden zusätzlichen Optionen sind für iSCSI nur:

| Option | Beschreibung |
|------------------------------|--|
| <code>fileSystemType</code> | Legt das Dateisystem fest, das zum Formatieren von iSCSI-Volumes verwendet wird. Die Standardeinstellung lautet <code>ext4</code> . Gültige Werte sind <code>ext3</code> , <code>ext4</code> , und <code>xf</code> s. |
| <code>spaceAllocation</code> | Einstellen auf <code>false</code> Deaktiviert die Funktion zur Speicherplatzzuweisung der LUN. Der Standardwert ist <code>true</code> , Die Bedeutung von ONTAP benachrichtigt den Host, wenn der Speicherplatz des Volume erschöpft ist, und die LUN im Volume kann Schreibvorgänge nicht akzeptieren. Mit dieser Option kann ONTAP auch automatisch Speicherplatz freigeben, wenn der Host Daten löscht. |

Beispiele

Sehen Sie sich die folgenden Beispiele an:

- 10 gib-Volume erstellen:

```
docker volume create -d netapp --name demo -o size=10G -o
encryption=true
```

- Erstellen Sie ein 100 gib Volume mit Snapshots:

```
docker volume create -d netapp --name demo -o size=100G -o
snapshotPolicy=default -o snapshotReserve=10
```

- Erstellen Sie ein Volume, bei dem das setuid-Bit aktiviert ist:

```
docker volume create -d netapp --name demo -o unixPermissions=4755
```

Die minimale Volume-Größe beträgt 20 MiB.

Wenn die Snapshot Reserve nicht angegeben wird und die Snapshot-Richtlinie lautet `none`, Trident wird eine Snapshot-Reserve von 0% verwenden.

- Erstellung eines Volumes ohne Snapshot-Richtlinie und ohne Snapshot-Reserve:

```
docker volume create -d netapp --name my_vol --opt snapshotPolicy=none
```

- Erstellen Sie ein Volume ohne Snapshot-Richtlinie und eine individuelle Snapshot-Reserve von 10 %:

```
docker volume create -d netapp --name my_vol --opt snapshotPolicy=none
--opt snapshotReserve=10
```

- Erstellen Sie ein Volume mit einer Snapshot-Richtlinie und einer individuellen Snapshot-Reserve von 10 %:

```
docker volume create -d netapp --name my_vol --opt
snapshotPolicy=myPolicy --opt snapshotReserve=10
```

- Erstellen Sie ein Volume mit einer Snapshot-Richtlinie, und akzeptieren Sie die standardmäßige Snapshot-Reserve von ONTAP (normalerweise 5%):

```
docker volume create -d netapp --name my_vol --opt
snapshotPolicy=myPolicy
```

Element Software-Volume-Optionen

Die Element Softwareoptionen bieten Zugriff auf die Größe und Quality of Service (QoS)-Richtlinien für das

Volume. Beim Erstellen des Volumes wird die ihr zugeordnete QoS-Richtlinie mithilfe des festgelegt `-o type=service_level` Terminologie

Der erste Schritt bei der Definition eines QoS-Service-Levels mit Element driver besteht darin, mindestens einen Typ zu erstellen und die minimalen, maximalen und Burst-IOPS anzugeben, die mit einem Namen in der Konfigurationsdatei verbunden sind.

Darüber anderem sind bei Volumes für Element Software folgende Optionen verfügbar:

| Option | Beschreibung |
|-----------|--|
| size | Die Größe des Volumens, standardmäßig auf 1gib oder Konfigurationseintrag... "Standardwerte": {"Größe": "5G"}. |
| blocksize | Verwenden Sie entweder 512 oder 4096, standardmäßig 512 oder den Konfigurationseintrag StandardBlockSize. |

Beispiel

In der folgenden Beispielkonfigurationsdatei finden Sie QoS-Definitionen:

```

{
  "...": "...",
  "Types": [
    {
      "Type": "Bronze",
      "Qos": {
        "minIOPS": 1000,
        "maxIOPS": 2000,
        "burstIOPS": 4000
      }
    },
    {
      "Type": "Silver",
      "Qos": {
        "minIOPS": 4000,
        "maxIOPS": 6000,
        "burstIOPS": 8000
      }
    },
    {
      "Type": "Gold",
      "Qos": {
        "minIOPS": 6000,
        "maxIOPS": 8000,
        "burstIOPS": 10000
      }
    }
  ]
}

```

In der obigen Konfiguration haben wir drei Richtliniendefinitionen: Bronze, Silver und Gold. Diese Namen sind frei wählbar.

- Erstellen eines 10 gib Gold-Volumes:

```
docker volume create -d solidfire --name sfGold -o type=Gold -o size=10G
```

- Erstellen eines 100 gib Bronze-Volumens:

```
docker volume create -d solidfire --name sfBronze -o type=Bronze -o
size=100G
```


Sammelt Protokolle

Sie können Protokolle erfassen, um Hilfe bei der Fehlerbehebung zu erhalten. Die Methode zur Erfassung der Protokolle variiert je nach Ausführung des Docker Plug-ins.

Sammelt Protokolle für die Fehlerbehebung

Schritte

1. Wenn Sie Astra Trident mit der empfohlenen Managed Plugin-Methode ausführen (d. h. mit `docker plugin` Befehle), zeigen Sie sie wie folgt an:

```
docker plugin ls
ID                NAME                DESCRIPTION
ENABLED
4fb97d2b956b     netapp:latest      nDVP - NetApp Docker Volume
Plugin    false
journalctl -u docker | grep 4fb97d2b956b
```

Die Standardprotokollierungsebene sollte Ihnen die Diagnose der meisten Probleme ermöglichen. Wenn Sie feststellen, dass das nicht genug ist, können Sie Debug-Protokollierung aktivieren.

2. Um die Debug-Protokollierung zu aktivieren, installieren Sie das Plugin mit aktivierter Debug-Protokollierung:

```
docker plugin install netapp/trident-plugin:<version> --alias <alias>
debug=true
```

Oder aktivieren Sie Debug-Protokollierung, wenn das Plugin bereits installiert ist:

```
docker plugin disable <plugin>
docker plugin set <plugin> debug=true
docker plugin enable <plugin>
```

3. Wenn Sie die Binärdatei selbst auf dem Host ausführen, sind Protokolle auf den Hosts verfügbar `/var/log/netappdvp` Verzeichnis. Um die Debug-Protokollierung zu aktivieren, geben Sie an `-debug` Wenn Sie das Plugin ausführen.

Allgemeine Tipps zur Fehlerbehebung

- Das häufigste Problem, in dem neue Benutzer auftreten, ist eine fehlerhafte Konfiguration, die verhindert, dass das Plugin initialisiert wird. Wenn dies geschieht, werden Sie wahrscheinlich eine Meldung wie diese sehen, wenn Sie versuchen, das Plugin zu installieren oder zu aktivieren:

```
Error response from daemon: dial unix /run/docker/plugins/<id>/netapp.sock:
connect: no such file or directory
```

Das bedeutet, dass das Plugin nicht gestartet werden konnte. Zum Glück wurde das Plugin mit einer umfassenden Protokollierungsfunktion aufgebaut, die Ihnen bei der Diagnose der meisten Probleme helfen sollte, die Sie wahrscheinlich auftreten.

- Bei Problemen mit der Montage eines PV in einem Behälter, darauf achten `rpcbind` Wird installiert und ausgeführt. Verwenden Sie den erforderlichen Paket-Manager für das Host-Betriebssystem, und überprüfen Sie, ob `rpcbind` Wird ausgeführt. Sie können den Status des `rpcbind`-Dienstes überprüfen, indem Sie ein ausführen `systemctl status rpcbind` Oder gleichwertige Informationen.

Management mehrerer Astra Trident Instanzen

Wenn mehrere Storage-Konfigurationen gleichzeitig verfügbar sind, sind mehrere Instanzen von Trident erforderlich. Der Schlüssel für mehrere Instanzen besteht darin, ihnen verschiedene Namen mit dem zu geben `--alias` Sie haben die Option mit dem Container-Plug-in oder `--volume-driver` Option beim Instanzieren von Trident auf dem Host.

Schritte für Docker Managed Plug-in (Version 1.13/17.03 oder höher)

1. Starten Sie die erste Instanz, die einen Alias und eine Konfigurationsdatei angibt.

```
docker plugin install --grant-all-permissions --alias silver
netapp/trident-plugin:21.07 config=silver.json
```

2. Starten Sie die zweite Instanz, indem Sie einen anderen Alias und eine andere Konfigurationsdatei angeben.

```
docker plugin install --grant-all-permissions --alias gold
netapp/trident-plugin:21.07 config=gold.json
```

3. Erstellen Sie Volumes, die den Alias als Treibername angeben.

Beispiel für Gold Volume:

```
docker volume create -d gold --name ntapGold
```

Beispiel für Silbervolumen:

```
docker volume create -d silver --name ntapSilver
```

Schritte für herkömmliche (Version 1.12 oder früher)

1. Starten Sie das Plug-in mit einer NFS-Konfiguration unter Verwendung einer benutzerdefinierten Treiber-ID:

```
sudo trident --volume-driver=netapp-nas --config=/path/to/config
-nfs.json
```

2. Starten Sie das Plug-in mit einer iSCSI-Konfiguration unter Verwendung einer benutzerdefinierten Treiber-ID:

```
sudo trident --volume-driver=netapp-san --config=/path/to/config
-iscsi.json
```

3. Stellen Sie Docker Volumes für jede Treiberinstanz bereit:

Zum Beispiel für NFS:

```
docker volume create -d netapp-nas --name my_nfs_vol
```

Beispiel für iSCSI:

```
docker volume create -d netapp-san --name my_iscsi_vol
```

Optionen für die Storage-Konfiguration

Sehen Sie sich die Konfigurationsoptionen der Astra Trident Konfigurationen an.

Globale Konfigurationsoptionen

Diese Konfigurationsoptionen sind für alle Astra Trident Konfigurationen anwendbar, unabhängig von der genutzten Storage-Plattform.

| Option | Beschreibung | Beispiel |
|-------------------|---|---|
| version | Versionsnummer der Konfigurationsdatei | 1 |
| storageDriverName | Name des Speichertreibers | ontap-nas, ontap-san, ontap-nas-economy, ontap-nas-flexgroup, solidfire-san |
| storagePrefix | Optionales Präfix für Volumen-Namen Standard: netappdvp_. | staging_ |

| Option | Beschreibung | Beispiel |
|-----------------|---|----------|
| limitVolumeSize | Optionale Einschränkung von Volume-Größen. Standard: „“ (nicht erzwungen) | 10g |



Verwenden Sie es nicht `storagePrefix` (Einschließlich Standard) für Element-Back-Ends. Standardmäßig wird der verwendet `solidfire-san` Der Treiber ignoriert diese Einstellung und verwendet kein Präfix. Wir empfehlen die Verwendung einer bestimmten TenantID für die Docker Volume-Zuordnung oder die Verwendung der Attributdaten, die mit der Docker-Version, den Treiber-Informationen und dem Raw-Namen aus Docker gefüllt sind, in Fällen, in denen Namensnennung verwendet wurde.

Es stehen Standardoptionen zur Verfügung, damit Sie sie nicht für jedes erstellte Volume angeben müssen. Die Option `size` ist für alle Controller-Typen verfügbar. Im Abschnitt zur ONTAP-Konfiguration finden Sie ein Beispiel dafür, wie Sie die Standard-Volume-Größe festlegen.

| Option | Beschreibung | Beispiel |
|--------|--|----------|
| size | Optionale Standardgröße für neue Volumes. Standard: 1G | 10G |

ONTAP-Konfiguration

Zusätzlich zu den oben genannten globalen Konfigurationswerten stehen bei Verwendung von ONTAP folgende Optionen auf oberster Ebene zur Verfügung.

| Option | Beschreibung | Beispiel |
|---------------|---|----------|
| managementLIF | IP-Adresse des ONTAP Management LIF. Sie können einen vollqualifizierten Domänennamen (FQDN) angeben. | 10.0.0.1 |

| Option | Beschreibung | Beispiel |
|---------------------|---|----------|
| dataLIF | <p>IP-Adresse des LIF-Protokolls.</p> <p>ONTAP NAS drivers: Wir empfehlen die Angabe <code>dataLIF</code>. Falls nicht vorgesehen, ruft Astra Trident Daten-LIFs von der SVM ab. Sie können einen vollständig qualifizierten Domännennamen (FQDN) angeben, der für die NFS-Mount-Vorgänge verwendet werden soll. Damit können Sie ein Round-Robin-DNS zum Load-Balancing über mehrere Daten-LIFs erstellen.</p> <p>ONTAP-SAN-Treiber: Geben Sie nicht für iSCSI an. Astra Trident verwendet "ONTAP selektive LUN-Zuordnung" Um die iSCSI LIFs zu ermitteln, die für die Einrichtung einer Multi-Path-Sitzung erforderlich sind. Wenn eine Warnung erzeugt wird <code>dataLIF</code> ist explizit definiert.</p> | 10.0.0.2 |
| svm | Storage Virtual Machine zu verwenden (erforderlich, falls Management LIF eine Cluster-LIF ist) | svm_nfs |
| username | Benutzername zur Verbindung mit dem Speichergerät | vsadmin |
| password | Passwort für die Verbindung mit dem Speichergerät | secret |
| aggregate | Aggregat für die Bereitstellung (optional, wenn eingestellt, muss der SVM zugewiesen werden) Für das <code>ontap-nas-flexgroup</code> Treiber, diese Option wird ignoriert. Alle der SVM zugewiesenen Aggregate werden zur Bereitstellung eines FlexGroup Volumes verwendet. | aggr1 |
| limitAggregateUsage | Optionale, fail-Provisioning-Funktion, wenn die Nutzung über diesem Prozentsatz liegt | 75% |

| Option | Beschreibung | Beispiel |
|------------------|--|--------------|
| nfsMountOptions | Feingranulare Steuerung der NFS-Mount-Optionen; standardmäßig „-o nfsvers=3“. Nur für die verfügbar ontap-nas Und ontap-nas-economy Fahrer. "Siehe Informationen zur NFS-Host-Konfiguration hier" . | -o nfsvers=4 |
| igroupName | Astra Trident erstellt und managt pro Node igroups Als netappdvp. Dieser Wert kann nicht geändert oder weggelassen werden. Nur für die verfügbar ontap-san Fahrer. | netappdvp |
| limitVolumeSize | Maximale anforderbare Volume-Größe und übergeordnete qtree Volume-Größe * Für die ontap-nas-economy Treiber, diese Option schränkt zusätzlich die Größe der FlexVols ein, die es erstellt*. | 300g |
| qtreesPerFlexvol | Maximale Anzahl der qtrees pro FlexVol, die im Bereich [50, 300] liegen müssen, die Standardeinstellung ist 200. * Für die ontap-nas-economy Treiber: Mit dieser Option kann die maximale Anzahl von qtrees pro FlexVol* angepasst werden. | 300 |

Es stehen Standardoptionen zur Verfügung, um zu vermeiden, dass sie auf jedem von Ihnen erstellten Volume angegeben werden müssen:

| Option | Beschreibung | Beispiel |
|----------------|--|----------|
| spaceReserve | Modus für Speicherplatzreservierung; none (Thin Provisioning) oder volume (Dick) | none |
| snapshotPolicy | Zu verwendende Snapshot-Richtlinie, Standard ist none | none |

| Option | Beschreibung | Beispiel |
|-----------------|---|----------------------|
| snapshotReserve | Der Prozentsatz der Snapshot-Reserve ist standardmäßig „“, um den ONTAP-Standardwert zu akzeptieren | 10 |
| splitOnClone | Teilen Sie einen Klon bei der Erstellung von seinem übergeordneten Element auf. Dies ist standardmäßig der Standardwert <code>false</code> | <code>false</code> |
| encryption | <p>Aktiviert NetApp Volume Encryption (NVE) auf dem neuen Volume, standardmäßig aktiviert <code>false</code>. NVE muss im Cluster lizenziert und aktiviert sein, damit diese Option verwendet werden kann.</p> <p>Wenn NAE auf dem Backend aktiviert ist, wird jedes im Astra Trident bereitgestellte Volume NAE aktiviert.</p> <p>Weitere Informationen finden Sie unter: "Astra Trident arbeitet mit NVE und NAE zusammen".</p> | Richtig |
| unixPermissions | NAS-Option für bereitgestellte NFS-Volumes, standardmäßig auf <code>777</code> | 777 |
| snapshotDir | NAS-Option für den Zugriff auf die <code>.snapshot</code> Verzeichnis, Standardeinstellung <code>false</code> | <code>true</code> |
| exportPolicy | NAS-Option für die zu verwendende NFS-Exportrichtlinie, standardmäßig auf <code>default</code> | <code>default</code> |
| securityStyle | <p>NAS-Option für Zugriff auf das bereitgestellte NFS-Volume.</p> <p>NFS unterstützt <code>mixed</code> Und <code>unix</code> Sicherheitsstile. Die Standardeinstellung lautet <code>unix</code>.</p> | <code>unix</code> |

| Option | Beschreibung | Beispiel |
|-----------------------------|---|-------------------|
| <code>fileSystemType</code> | SAN-Option zum Auswählen des Dateisystemtyps, standardmäßig auf <code>ext4</code> | <code>xf</code> s |
| <code>tieringPolicy</code> | Zu verwendende Tiering-Richtlinie, Standard ist <code>none</code> ; <code>snapshot-only</code> Für Konfiguration vor ONTAP 9.5 SVM-DR | <code>none</code> |

Skalierungsoptionen

Der `ontap-nas` Und `ontap-san` Treiber erstellen für jedes Docker Volume eine ONTAP FlexVol. ONTAP unterstützt bis zu 1000 FlexVols pro Cluster Node mit einem Cluster maximal 12,000 FlexVols. Wenn die Anforderungen für das Docker Volume diesen Anforderungen entsprechen, wird der angezeigt `ontap-nas` Aufgrund der zusätzlichen Funktionen von FlexVols, wie dem granularen Docker-Volume-Snapshot und Klonen, ist der Treiber die bevorzugte NAS-Lösung.

Wenn Sie mehr Docker Volumes benötigen, als durch die FlexVol-Limits unterstützt werden können, wählen Sie die Option `ontap-nas-economy` Oder im `ontap-san-economy` Treiber.

Der `ontap-nas-economy` Der Treiber erstellt Docker Volumes als ONTAP qtrees innerhalb eines Pools automatisch verwalteter FlexVols. Qtrees bieten eine wesentlich größere Skalierung – bis zu 100,000 pro Cluster-Node und 2,400,000 pro Cluster – zu Lasten einiger Funktionen. Der `ontap-nas-economy` Der Treiber unterstützt keine granularen Snapshots oder Klone von Docker Volumes.



Der `ontap-nas-economy` Treiber wird derzeit in Docker Swarm nicht unterstützt, da Swarm die Volume-Erstellung nicht über mehrere Nodes hinweg orchestriert.

Der `ontap-san-economy` Der Treiber erstellt Docker Volumes als ONTAP LUNs in einem gemeinsamen Pool automatisch verwalteter FlexVols. Somit ist jede FlexVol nicht auf nur eine LUN beschränkt und bietet eine bessere Skalierbarkeit für SAN-Workloads. Je nach Storage Array unterstützt ONTAP bis zu 16384 LUNs pro Cluster. Da es sich bei den Volumes um LUNs handelt, unterstützt dieser Treiber granulare Docker Snapshots und Klone.

Wählen Sie die aus `ontap-nas-flexgroup` Treiber zur Erhöhung der Parallelität zu einem einzelnen Volume, das mit Milliarden von Dateien im Petabyte-Bereich wachsen kann. Zu den idealen Anwendungsfällen für FlexGroups gehören KI/ML/DL, Big Data und Analysen, Softwareentwicklung, Streaming, Datei-Repositorys und so weiter. Trident verwendet alle Aggregate, die einer SVM bei der Bereitstellung eines FlexGroup-Volumes zugewiesen sind. Die Unterstützung von FlexGroup in Trident muss darüber hinaus Folgendes beachtet werden:

- ONTAP Version 9.2 oder höher erforderlich.
- Ab diesem Text unterstützt FlexGroups nur NFS v3.
- Empfohlen, die 64-Bit-NFSv3-IDs für die SVM zu aktivieren.
- Die empfohlene minimale FlexGroup-Größe beträgt 100 GB.
- Klonen wird für FlexGroup Volumes nicht unterstützt.

Informationen zu FlexGroups und Workloads, die für FlexGroups geeignet sind, finden Sie im ["NetApp](#)

FlexGroup Volume Best Practices und Implementierungsleitfaden".

Um erweiterte Funktionen und die enorme Skalierbarkeit in derselben Umgebung zu erhalten, können Sie mehrere Instanzen des Docker Volume Plug-ins ausführen. Dabei kommt ein Storage-Plug-in zum Einsatz `ontap-nas` Und ein anderes mit `ontap-nas-economy`.

Beispiel für ONTAP-Konfigurationsdateien

NFS Beispiel für `ontap-nas` Fahrer

```
{
  "version": 1,
  "storageDriverName": "ontap-nas",
  "managementLIF": "10.0.0.1",
  "dataLIF": "10.0.0.2",
  "svm": "svm_nfs",
  "username": "vsadmin",
  "password": "password",
  "aggregate": "aggr1",
  "defaults": {
    "size": "10G",
    "spaceReserve": "none",
    "exportPolicy": "default"
  }
}
```

NFS Beispiel für `ontap-nas-flexgroup` Fahrer

```
{
  "version": 1,
  "storageDriverName": "ontap-nas-flexgroup",
  "managementLIF": "10.0.0.1",
  "dataLIF": "10.0.0.2",
  "svm": "svm_nfs",
  "username": "vsadmin",
  "password": "password",
  "defaults": {
    "size": "100G",
    "spaceReserve": "none",
    "exportPolicy": "default"
  }
}
```

NFS Beispiel für `ontap-nas-economy` Fahrer

```

{
  "version": 1,
  "storageDriverName": "ontap-nas-economy",
  "managementLIF": "10.0.0.1",
  "dataLIF": "10.0.0.2",
  "svm": "svm_nfs",
  "username": "vsadmin",
  "password": "password",
  "aggregate": "aggr1"
}

```

ISCSI-Beispiel für ontap-san Fahrer

```

{
  "version": 1,
  "storageDriverName": "ontap-san",
  "managementLIF": "10.0.0.1",
  "dataLIF": "10.0.0.3",
  "svm": "svm_iscsi",
  "username": "vsadmin",
  "password": "password",
  "aggregate": "aggr1",
  "igroupName": "netappdvp"
}

```

NFS Beispiel für ontap-san-economy Fahrer

```

{
  "version": 1,
  "storageDriverName": "ontap-san-economy",
  "managementLIF": "10.0.0.1",
  "dataLIF": "10.0.0.3",
  "svm": "svm_iscsi_eco",
  "username": "vsadmin",
  "password": "password",
  "aggregate": "aggr1",
  "igroupName": "netappdvp"
}

```

Konfiguration von Element Software

Zusätzlich zu den Werten einer globalen Konfiguration sind bei Verwendung von Element Software (NetApp HCI/SolidFire) diese Optionen verfügbar.

| Option | Beschreibung | Beispiel |
|------------------|--|--|
| Endpoint | \Https://<login>:<password>@<mvi p>/json-rpc/<element-version> | https://admin:admin@192.168.160.3/json-rpc/8.0 |
| SVIP | ISCSI-IP-Adresse und -Port | 10.0.0.7:3260 Uhr |
| TenantName | SolidFireF Mandanten zu verwenden (erstellt, falls nicht gefunden) | docker |
| InitiatorIFace | Geben Sie die Schnittstelle an, wenn der iSCSI-Datenverkehr auf eine nicht-Standardschnittstelle beschränkt wird | default |
| Types | QoS-Spezifikationen | Siehe das Beispiel unten |
| LegacyNamePrefix | Präfix für aktualisierte Trident Installationen. Wenn Sie eine Version von Trident vor 1.3.2 verwendet und ein Upgrade mit vorhandenen Volumes durchführen, müssen Sie diesen Wert einstellen, um auf die alten Volumes zuzugreifen, die über die Volume-Name-Methode zugeordnet wurden. | netappdvp- |

Der `solidfire-san` Der Treiber unterstützt Docker Swarm nicht.

Beispiel für eine Konfigurationsdatei für die Element Software

```

{
  "version": 1,
  "storageDriverName": "solidfire-san",
  "Endpoint": "https://admin:admin@192.168.160.3/json-rpc/8.0",
  "SVIP": "10.0.0.7:3260",
  "TenantName": "docker",
  "InitiatorIFace": "default",
  "Types": [
    {
      "Type": "Bronze",
      "Qos": {
        "minIOPS": 1000,
        "maxIOPS": 2000,
        "burstIOPS": 4000
      }
    },
    {
      "Type": "Silver",
      "Qos": {
        "minIOPS": 4000,
        "maxIOPS": 6000,
        "burstIOPS": 8000
      }
    },
    {
      "Type": "Gold",
      "Qos": {
        "minIOPS": 6000,
        "maxIOPS": 8000,
        "burstIOPS": 10000
      }
    }
  ]
}

```

Bekannte Probleme und Einschränkungen

Hier finden Sie Informationen zu bekannten Problemen und Einschränkungen bei der Verwendung von Astra Trident mit Docker.

Das Upgrade des Trident Docker Volume Plug-ins auf 20.10 und höher aus älteren Versionen führt zu einem Upgrade-Fehler, ohne dass solche Datei- oder Verzeichnisfehler auftreten.

Behelfslösung

1. Deaktivieren Sie das Plugin.

```
docker plugin disable -f netapp:latest
```

2. Entfernen Sie das Plug-in.

```
docker plugin rm -f netapp:latest
```

3. Installieren Sie das Plug-in neu, indem Sie das Zusatzmodul bereitstellen `config` Parameter.

```
docker plugin install netapp/trident-plugin:20.10 --alias netapp --grant  
-all-permissions config=config.json
```

Volume-Namen müssen mindestens 2 Zeichen lang sein.



Dies ist eine Docker-Client-Einschränkung. Der Client interpretiert einen einzelnen Zeichennamen als Windows-Pfad. "[Siehe Bug 25773](#)".

Docker Swarm hat bestimmte Verhaltensweisen, die Astra Trident nicht durch jede Storage- und Treiberkombination unterstützen können.

- Docker Swarm verwendet derzeit Volume-Namen anstelle der Volume-ID als eindeutige Volume-Kennung.
- Volume-Anforderungen werden gleichzeitig an jeden Node in einem Swarm-Cluster gesendet.
- Volume-Plug-ins (einschließlich Astra Trident) müssen auf jedem Knoten in einem Swarm-Cluster unabhängig ausgeführt werden.
Aufgrund der Art und Weise, wie ONTAP funktioniert und wie die `ontap-nas` und `ontap-san` Treiber funktionieren, sind sie die einzigen, die zufällig in der Lage, innerhalb dieser Einschränkungen zu arbeiten.

Der Rest der Fahrer unterliegt Themen wie Rennbedingungen, die dazu führen können, dass eine große Anzahl von Volumes für eine einzelne Anfrage ohne einen klaren „Gewinner“ erstellt werden; zum Beispiel hat Element eine Funktion, die es Volumes erlaubt, den gleichen Namen, aber unterschiedliche IDs zu haben.

NetApp hat das Docker-Team Feedback gegeben, lässt aber keinen Anzeichen für einen zukünftigen Regressanspruch haben.

Wenn eine FlexGroup bereitgestellt wird, stellt ONTAP keine zweite FlexGroup bereit, wenn die zweite FlexGroup über einen oder mehrere Aggregate verfügt, die mit der bereitgestellten FlexGroup gemeinsam genutzt werden.

Best Practices und Empfehlungen

Einsatz

Nutzen Sie bei der Implementierung von Astra Trident die hier aufgeführten Empfehlungen.

Implementieren Sie diesen in einem dedizierten Namespace

"Namespaces" Trennung zwischen verschiedenen Applikationen und gemeinsame Nutzung von Ressourcen gehören zu den Hinderungsgrund. Beispielsweise kann eine PVC aus einem Namespace nicht von einem anderen genutzt werden. Astra Trident stellt allen Namespaces im Kubernetes-Cluster PV-Ressourcen zur Verfügung und nutzt daher ein Service-Konto mit erhöhten Rechten.

Außerdem kann der Zugriff auf den Trident Pod dazu führen, dass Benutzer auf die Anmeldedaten des Storage-Systems und andere sensible Informationen zugreifen können. Es ist wichtig, dass Applikationsbenutzer und Management-Applikationen nicht in der Lage sind, auf die Trident Objektdefinitionen oder Pods selbst zuzugreifen.

Verwenden Sie Kontingente und Bereichsgrenzen, um den Storage-Verbrauch zu kontrollieren

Kubernetes bietet zusammen zwei Funktionen, die einen leistungsstarken Mechanismus zur Begrenzung des Ressourcenverbrauchs durch Applikationen bieten. Der **"Mechanismus für Storage-Kontingente"** Er ermöglicht dem Administrator, globale und Storage-klassenspezifische Verbrauchslimits für Kapazität und Objektanzahl pro Namespace zu implementieren. Außerdem mit A **"Bereichsgrenze"** Gewährleistet, dass die PVC-Anforderungen sowohl den minimalen als auch den maximalen Wert haben, bevor die Anforderung an die provisionierung weitergeleitet wird.

Diese Werte werden pro Namespace definiert, was bedeutet, dass jeder Namespace Werte definiert haben sollte, die ihren Ressourcenanforderungen entsprechen. Informationen dazu finden Sie hier **"Wie man Quoten nutzt"**.

Storage-Konfiguration

Jede Storage-Plattform im NetApp Portfolio verfügt über einzigartige Funktionen für Applikationen, die in Containern oder nicht unterstützt werden.

Plattformübersicht

Trident funktioniert mit ONTAP und Element. Es gibt keine Plattform, die besser für alle Anwendungen und Szenarien geeignet ist als die andere, aber bei der Auswahl einer Plattform sollten die Anforderungen der Anwendung und des Teams, das das Gerät verwaltet, berücksichtigt werden.

Sie sollten die Best Practices für das Host-Betriebssystem anhand des von Ihnen verwendeten Protokolls befolgen. Optional können Sie möglicherweise erwägen, falls verfügbar Best Practices für Applikationen mit Back-End-, Storage-Klassen- und PVC-Einstellungen zu integrieren, um den Storage für bestimmte Applikationen zu optimieren.

Best Practices für ONTAP und Cloud Volumes ONTAP

Best Practices zur Konfiguration von ONTAP und Cloud Volumes ONTAP für Trident enthalten.

Die folgenden Empfehlungen sind Richtlinien zur Konfiguration von ONTAP für Container-Workloads, die Volumes nutzen, die von Trident dynamisch bereitgestellt werden. Jeder sollte in Betracht gezogen und auf Angemessenheit in Ihrer Umgebung überprüft werden.

Verwenden Sie SVM(s) dediziert für Trident

Storage Virtual Machines (SVMs) sorgen für die Trennung von Mandanten auf einem ONTAP System. Durch die Zuweisung einer SVM für Applikationen können Berechtigungen delegation werden. Zudem lassen sich Best Practices anwenden, um den Ressourcenverbrauch zu begrenzen.

Für das Management der SVM sind verschiedene Optionen verfügbar:

- Stellen Sie die Cluster-Managementoberfläche in der Backend-Konfiguration zusammen mit entsprechenden Zugangsdaten bereit und geben Sie den SVM-Namen an.
- Erstellen Sie mit ONTAP System Manager oder der CLI eine dedizierte Managementoberfläche für die SVM.
- Teilen Sie die Managementrolle mit einer NFS-Datenschnittstelle.

In jedem Fall sollte sich die Schnittstelle im DNS enthalten, und beim Konfigurieren von Trident sollte der DNS-Name verwendet werden. Dadurch lassen sich einige DR-Szenarien, beispielsweise SVM-DR, vereinfachen, ohne die Aufbewahrung der Netzwerkidentität zu nutzen.

Es besteht keine Präferenz zwischen einer dedizierten oder gemeinsam genutzten Management-LIF für die SVM. Sie sollten jedoch sicherstellen, dass Ihre Netzwerksicherheitsrichtlinien mit dem von Ihnen gewählten Ansatz abgestimmt sind. Unabhängig davon sollte die Management-LIF über DNS zugänglich sein, um ein Maximum an Flexibilität zu ermöglichen "SVM-DR" Zusammen mit Trident verwendet werden.

Begrenzung der maximalen Volume-Anzahl

ONTAP Storage-Systeme besitzen eine maximale Anzahl an Volumes, die je nach Softwareversion und Hardwareplattform unterschiedlich sind. Siehe "[NetApp Hardware Universe](#)" Für Ihre spezifische Plattform und ONTAP-Version, um die genauen Grenzen zu bestimmen. Wenn die Anzahl der Volumes erschöpft ist, schlägt die Bereitstellung nicht nur für Trident fehl, sondern für alle Storage-Anforderungen.

Trident `ontap-nas` Und `ontap-san` Treiber stellen für jedes erstellte Kubernetes Persistent Volume (PV) ein FlexVol Volume bereit. Der `ontap-nas-economy` Der Treiber erstellt ca. ein FlexVolum für alle 200 PVS (konfigurierbar zwischen 50 und 300). Der `ontap-san-economy` Treiber erstellt ca. ein FlexVolume für je 100 PVs (konfigurierbar zwischen 50 und 200). Damit Trident nicht alle verfügbaren Volumes im Storage-System verbraucht, sollten Sie ein Limit für die SVM festlegen. Dies können Sie über die Befehlszeile ausführen:

```
vserver modify -vserver <svm_name> -max-volumes <num_of_volumes>
```

Der Wert für `max-volumes` Variiert basierend auf verschiedenen für Ihre Umgebung spezifischen Kriterien:

- Die Anzahl der vorhandenen Volumes im ONTAP Cluster
- Die Anzahl der Volumes, die für andere Applikationen außerhalb von Trident bereitgestellt werden
- Die Anzahl der persistenten Volumes, die von Kubernetes-Applikationen genutzt werden sollen

Der `max-volumes` Der Wert sind die gesamten Volumes, die über alle Nodes im ONTAP Cluster bereitgestellt werden, und nicht über einen einzelnen ONTAP Node. Aus diesem Grund treten möglicherweise einige Bedingungen auf, bei denen auf einem ONTAP Cluster-Node mehr oder weniger mit Trident bereitgestellte Volumes als ein anderer Node vorhanden sind.

So kann beispielsweise ein ONTAP Cluster mit zwei Nodes maximal 2000 FlexVols hosten. Eine auf 1250 eingestellte maximale Volumenzahl erscheint sehr vernünftig. Wenn auch nur "[Aggregate](#)" Von einem Node wird der SVM zugewiesen. Oder die von einem Node zugewiesenen Aggregate können nicht bereitgestellt werden (z. B. aufgrund der Kapazität), dann wird der andere Node Ziel für alle mit Trident bereitgestellten Volumes. Das bedeutet, dass vor dem das Volume-Limit für diesen Node erreicht werden kann `max-volumes` Der Wert wird erreicht, was sowohl Trident als auch andere Volume-Vorgänge, die diesen Node verwenden, beeinträchtigt. **Diese Situation kann vermieden werden, indem sichergestellt wird, dass die Aggregate von jedem Node im Cluster der von Trident verwendeten SVM in gleicher Anzahl zugewiesen werden.**

Begrenzung der maximalen Größe der durch Trident erstellten Volumes

Verwenden Sie das, um die maximale Größe für Volumes zu konfigurieren, die mit Trident erstellt werden können `limitVolumeSize` Parameter in im `backend.json` Definition:

Neben der Kontrolle der Volume-Größe im Storage-Array sollten auch Kubernetes-Funktionen genutzt werden.

Trident für bidirektionales CHAP konfigurieren

Sie können in der Back-End-Definition den CHAP-Initiator und die Benutzernamen und Passwörter für das Ziel angeben und Trident CHAP auf der SVM aktivieren. Verwenden der `useCHAP` Parameter in der Back-End-Konfiguration authentifiziert Trident iSCSI-Verbindungen für ONTAP-Back-Ends mit CHAP.

Erstellen und Verwenden einer SVM QoS-Richtlinie

Die Nutzung einer ONTAP QoS-Richtlinie auf die SVM begrenzt die Anzahl der durch die von Trident bereitgestellten Volumes konsumierbaren IOPS. Dies hilft "[Verhindern Sie einen Schläger](#)" Oder nicht-kontrollierter Container, der Workloads außerhalb der Trident SVM beeinträchtigt.

Sie können in wenigen Schritten eine QoS-Richtlinie für die SVM erstellen. Die genauesten Informationen finden Sie in der Dokumentation Ihrer ONTAP-Version. Das folgende Beispiel erstellt eine QoS-Richtlinie, die die insgesamt für eine SVM verfügbaren IOPS auf 5000 begrenzt.

```
# create the policy group for the SVM
qos policy-group create -policy-group <policy_name> -vserver <svm_name>
-max-throughput 5000iops

# assign the policy group to the SVM, note this will not work
# if volumes or files in the SVM have existing QoS policies
vserver modify -vserver <svm_name> -qos-policy-group <policy_name>
```

Wenn zudem Ihre ONTAP Version sie unterstützt, können Sie den Einsatz eines minimalen QoS-Systems in Erwägung ziehen, um einen hohen Durchsatz für Container-Workloads zu gewährleisten. Die adaptive QoS ist nicht mit einer Richtlinie auf SVM-Ebene kompatibel.

Die Anzahl der für Container-Workloads dedizierten IOPS hängt von vielen Aspekten ab. Dazu zählen unter anderem:

- Anderen Workloads, die das Storage-Array nutzen Bei anderen Workloads, die nicht mit der Kubernetes-Implementierung zusammenhängen und die Storage-Ressourcen nutzen, sollte darauf achten, dass diese Workloads nicht versehentlich beeinträchtigt werden.
- Erwartete Workloads werden in Containern ausgeführt. Wenn Workloads mit hohen IOPS-Anforderungen in Containern ausgeführt werden, führt eine niedrige QoS-Richtlinie zu schlechten Erfahrungen.

Es muss daran erinnert werden, dass eine auf SVM-Ebene zugewiesene QoS-Richtlinie alle Volumes zur Verfügung hat, die der SVM bereitgestellt werden und sich denselben IOPS-Pool teilen. Wenn eine oder nur eine kleine Zahl von Container-Applikationen sehr hohe IOPS-Anforderungen erfüllen, kann dies zu einem problematischer für die anderen Container-Workloads werden. In diesem Fall empfiehlt es sich, QoS-Richtlinien pro Volume mithilfe von externer Automatisierung zuzuweisen.



Sie sollten die QoS Policy Group der SVM **only** zuweisen, wenn Ihre ONTAP Version älter als 9.8 ist.

Erstellen von QoS-Richtliniengruppen für Trident

Quality of Service (QoS) garantiert, dass die Performance kritischer Workloads nicht durch konkurrierende Workloads beeinträchtigt wird. ONTAP QoS-Richtliniengruppen bieten QoS-Optionen für Volumes und ermöglichen Benutzern, die Durchsatzgrenze für einen oder mehrere Workloads zu definieren. Weitere Informationen zur QoS finden Sie unter "[Garantierter Durchsatz durch QoS](#)".

Sie können QoS-Richtliniengruppen im Backend oder im Storage-Pool festlegen und werden auf jedes in diesem Pool oder Backend erstellte Volume angewendet.

ONTAP verfügt über zwei Arten von QoS-Richtliniengruppen: Herkömmliche und anpassungsfähige. Herkömmliche Richtliniengruppen bieten einen flachen maximalen Durchsatz (oder minimalen Durchsatz in späteren Versionen) in IOPS. Adaptive QoS skaliert den Durchsatz automatisch auf die Workload-Größe und erhält das Verhältnis von IOPS zu TB-fähigen GB-Werten, wenn sich die Workload-Größe ändert. Wenn Sie Hunderte oder Tausende Workloads in einer großen Implementierung managen, bietet sich somit ein erheblicher Vorteil.

Beachten Sie beim Erstellen von QoS-Richtliniengruppen Folgendes:

- Sie sollten die einstellen `qosPolicy` Taste im `defaults` Block der Back-End-Konfiguration. Im folgenden Back-End-Konfigurationsbeispiel:

```

---
version: 1
storageDriverName: ontap-nas
managementLIF: 0.0.0.0
dataLIF: 0.0.0.0
svm: svm0
username: user
password: pass
defaults:
  qosPolicy: standard-pg
storage:
- labels:
  performance: extreme
  defaults:
  adaptiveQosPolicy: extremely-adaptive-pg
- labels:
  performance: premium
  defaults:
  qosPolicy: premium-pg

```

- Sie sollten die Richtliniengruppen pro Volume anwenden, damit jedes Volume den gesamten von der Richtliniengruppe angegebenen Durchsatz erhält. Gemeinsame Richtliniengruppen werden nicht unterstützt.

Weitere Informationen zu QoS-Richtliniengruppen finden Sie unter ["ONTAP 9.8 QoS-Befehle"](#).

Beschränken Sie den Zugriff auf die Storage-Ressourcen auf Kubernetes-Cluster-Mitglieder

Der Zugriff auf die durch Trident erstellten NFS-Volumes und iSCSI-LUNs ist eine entscheidende Komponente der Sicherheit für die Kubernetes-Implementierung. Auf diese Weise wird verhindert, dass Hosts, die nicht zum Kubernetes Cluster gehören, auf die Volumes zugreifen und Daten unerwartet ändern können.

Es ist wichtig zu wissen, dass Namespaces die logische Grenze für Ressourcen in Kubernetes sind. Es wird angenommen, dass Ressourcen im selben Namespace gemeinsam genutzt werden können. Es gibt jedoch keine Cross-Namespace-Funktion. Dies bedeutet, dass PVS zwar globale Objekte sind, aber wenn sie an ein PVC gebunden sind, nur über Pods zugänglich sind, die sich im selben Namespace befinden. **Es ist wichtig sicherzustellen, dass Namensräume verwendet werden, um eine Trennung zu gewährleisten, wenn angemessen.**

Die meisten Unternehmen haben im Zusammenhang mit der Datensicherheit bei Kubernetes die Sorge, dass ein Container-Prozess auf den Storage zugreifen kann, der am Host gemountet ist; dieser ist jedoch nicht für den Container bestimmt. "[Namespaces](#)" wurden entwickelt, um eine solche Art von Kompromiss zu verhindern. Allerdings gibt es eine Ausnahme: Privilegierte Container.

Ein privilegierter Container ist ein Container, der mit wesentlich mehr Berechtigungen auf Hostebene als normal ausgeführt wird. Diese werden standardmäßig nicht verweigert. Daher sollten Sie diese Funktion mithilfe von deaktivieren "[Pod-Sicherheitsrichtlinien](#)".

Bei Volumes, für die der Zugriff von Kubernetes und externen Hosts gewünscht wird, sollte der Storage auf herkömmliche Weise gemanagt werden. Dabei wird das PV durch den Administrator eingeführt und nicht von

Trident gemanagt. So wird sichergestellt, dass das Storage Volume nur zerstört wird, wenn sowohl Kubernetes als auch externe Hosts getrennt haben und das Volume nicht mehr nutzen. Zusätzlich kann eine benutzerdefinierte Exportrichtlinie angewendet werden, die den Zugriff von den Kubernetes-Cluster-Nodes und Zielsevern außerhalb des Kubernetes-Clusters ermöglicht.

Für Bereitstellungen mit dedizierten Infrastruktur-Nodes (z. B. OpenShift) oder anderen Nodes, die Benutzerapplikationen nicht planen können, sollten separate Exportrichtlinien verwendet werden, um den Zugriff auf Speicherressourcen weiter zu beschränken. Dies umfasst die Erstellung einer Exportrichtlinie für Services, die auf diesen Infrastruktur-Nodes bereitgestellt werden (z. B. OpenShift Metrics and Logging Services), sowie Standardanwendungen, die auf nicht-Infrastruktur-Nodes bereitgestellt werden.

Verwenden Sie eine dedizierte Exportrichtlinie

Sie sollten sicherstellen, dass für jedes Backend eine Exportrichtlinie vorhanden ist, die nur den Zugriff auf die im Kubernetes-Cluster vorhandenen Nodes erlaubt. Trident kann Richtlinien für den Export automatisch erstellen und managen. So beschränkt Trident den Zugriff auf die Volumes, die ihm im Kubernetes Cluster zur Verfügung stehen, und vereinfacht das Hinzufügen/Löschen von Nodes.

Alternativ können Sie auch eine Exportrichtlinie manuell erstellen und mit einer oder mehreren Exportregeln füllen, die die Zugriffsanforderung für die einzelnen Knoten bearbeiten:

- Verwenden Sie die `vserver export-policy create` ONTAP CLI-Befehl zum Erstellen der Exportrichtlinie.
- Fügen Sie mit dem Regeln zur Exportrichtlinie hinzu `vserver export-policy rule create` ONTAP-CLI-Befehl.

Wenn Sie diese Befehle ausführen, können Sie die Zugriffsrechte der Kubernetes-Nodes auf die Daten beschränken.

Deaktivieren `showmount` Für die Applikations-SVM

Der `showmount` Mit dieser Funktion kann ein NFS-Client die SVM für eine Liste verfügbarer NFS-Exporte abfragen. Ein im Kubernetes-Cluster implementierter Pod kann die Ausgabe `showmount -e` Befehl mit der Daten-LIF und erhält eine Liste der verfügbaren Mounts, einschließlich derer, auf die es keinen Zugriff hat. Obwohl dies für sich kein Sicherheitskompromiss ist, stellt es keine unnötigen Informationen bereit, die einem nicht autorisierten Benutzer die Verbindung zu einem NFS-Export ermöglichen.

Sie sollten deaktivieren `showmount` Mithilfe des ONTAP-CLI-Befehls auf SVM-Ebene:

```
vserver nfs modify -vserver <svm_name> -showmount disabled
```

SolidFire Best Practices in sich vereint

Lesen Sie Best Practices zur Konfiguration von SolidFire Storage für Trident.

Erstellen Eines SolidFire-Kontos

Jedes SolidFire-Konto stellt einen eindeutigen Volume-Eigentümer dar und erhält seine eigenen Anmeldeinformationen für das Challenge-Handshake Authentication Protocol (CHAP). Sie können auf Volumes zugreifen, die einem Konto zugewiesen sind, entweder über den Kontonamen und die relativen CHAP-Anmeldeinformationen oder über eine Zugriffsgruppe für Volumes. Einem Konto können bis zu zweitausend Volumes zugewiesen sein, ein Volume kann jedoch nur zu einem Konto gehören.

Erstellen einer QoS-Richtlinie

Verwenden Sie QoS-Richtlinien (Quality of Service) von SolidFire, um eine standardisierte Quality of Service-Einstellung zu erstellen und zu speichern, die auf viele Volumes angewendet werden kann.

Sie können QoS-Parameter für einzelne Volumes festlegen. Die Performance für jedes Volume kann durch drei konfigurierbare Parameter bestimmt werden, die QoS definieren: Das IOPS-Minimum, das IOPS-Maximum und die Burst-IOPS.

Hier sind die möglichen Minimum-, Maximum- und Burst-IOPS für die 4-KB-Blockgröße.

| IOPS-Parameter | Definition | Mindestens Wert | Standardwert | Maximale Wert (4 KB) |
|----------------|--|-----------------|--------------|----------------------|
| IOPS-Minimum | Das garantierte Performance-Level für ein Volume | 50 | 50 | 15000 |
| IOPS-Maximum | Die Leistung überschreitet dieses Limit nicht. | 50 | 15000 | 200,000 |
| IOPS-Burst | Maximale IOPS in einem kurzen Burst-Szenario zulässig. | 50 | 15000 | 200,000 |



Obwohl die IOPS-Maximum und die Burst-IOPS so hoch wie 200,000 sind, wird die tatsächliche maximale Performance eines Volumes durch die Nutzung von Clustern und die Performance pro Node begrenzt.

Die Blockgröße und die Bandbreite haben einen direkten Einfluss auf die Anzahl der IOPS. Mit zunehmender Blockgröße erhöht das System die Bandbreite auf ein Niveau, das für die Verarbeitung größerer Blockgrößen erforderlich ist. Mit der steigenden Bandbreite sinkt auch die Anzahl an IOPS, die das System erreichen kann. Siehe "[SolidFire Quality of Service](#)" Weitere Informationen zu QoS und Performance.

SolidFire Authentifizierung

Element unterstützt zwei Authentifizierungsmethoden: CHAP und Volume Access Groups (VAG). CHAP verwendet das CHAP-Protokoll, um den Host am Backend zu authentifizieren. Volume Access Groups steuern den Zugriff auf die Volumes, die durch sie bereitgestellt werden. Da die Authentifizierung einfacher ist und über keine Grenzen für die Skalierung verfügt, empfiehlt NetApp die Verwendung von CHAP.



Trident mit dem erweiterten CSI-provisioner unterstützt die Verwendung von CHAP-Authentifizierung. Vags sollten nur im traditionellen nicht-CSI-Betriebsmodus verwendet werden.

CHAP-Authentifizierung (Verifizierung, dass der Initiator der vorgesehene Volume-Benutzer ist) wird nur mit der Account-basierten Zugriffssteuerung unterstützt. Wenn Sie CHAP zur Authentifizierung verwenden, stehen zwei Optionen zur Verfügung: Unidirektionales CHAP und bidirektionales CHAP. Unidirektionales CHAP authentifiziert den Volume-Zugriff mithilfe des SolidFire-Kontonamens und des Initiatorgeheimnisses. Die bidirektionale CHAP-Option bietet die sicherste Möglichkeit zur Authentifizierung des Volumes, da das Volume den Host über den Kontonamen und den Initiatorschlüssel authentifiziert und dann der Host das Volume über den Kontonamen und den Zielschlüssel authentifiziert.

Wenn CHAP jedoch nicht aktiviert werden kann und Vags erforderlich sind, erstellen Sie die Zugriffsgruppe und fügen Sie die Hostinitiatoren und Volumes der Zugriffsgruppe hinzu. Jeder IQN, den Sie einer Zugriffsgruppe hinzufügen, kann mit oder ohne CHAP-Authentifizierung auf jedes Volume in der Gruppe zugreifen. Wenn der iSCSI-Initiator für die Verwendung der CHAP-Authentifizierung konfiguriert ist, wird die kontenbasierte Zugriffssteuerung verwendet. Wenn der iSCSI-Initiator nicht für die Verwendung der CHAP-Authentifizierung konfiguriert ist, wird die Zugriffskontrolle für die Volume Access Group verwendet.

Wo finden Sie weitere Informationen?

Einige der Best Practices-Dokumentationen sind unten aufgeführt. Suchen Sie die ["NetApp Bibliothek"](#) Für die aktuellsten Versionen.

ONTAP

- ["NFS Best Practice- und Implementierungsleitfaden"](#)
- ["SAN-Administration-Leitfaden"](#) (Für iSCSI)
- ["iSCSI Express-Konfiguration für RHEL"](#)

Element Software

- ["Konfigurieren von SolidFire für Linux"](#)

NetApp HCI

- ["Voraussetzungen für die NetApp HCI-Implementierung"](#)
- ["Rufen Sie die NetApp Deployment Engine auf"](#)

Anwendung Best Practices Informationen

- ["Best Practices für MySQL auf ONTAP"](#)
- ["Best Practices für MySQL auf SolidFire"](#)
- ["NetApp SolidFire und Cassandra"](#)
- ["Best Practices für Oracle auf SolidFire"](#)
- ["Best Practices für PostgreSQL auf SolidFire"](#)

Nicht alle Applikationen haben spezifische Richtlinien. Daher ist es wichtig, mit Ihrem NetApp Team zusammenzuarbeiten und die darauf zu verwenden ["NetApp Bibliothek"](#) Und finden Sie die aktuellste Dokumentation.

Integration Von Astra Trident

Zur Integration von Astra Trident erfordern die folgenden Design- und Architekturelemente Integration: Treiberauswahl und -Implementierung, Storage-Class-Design, Virtual Pool Design, Persistent Volume Claim (PVC) Einfluss auf die Storage-Bereitstellung, auf den Volume-Betrieb und die OpenShift-Serviceimplementierung mit Astra Trident.

Auswahl und Implementierung der Treiber

Wählen Sie einen Back-End-Treiber für Ihr Speichersystem aus und implementieren Sie ihn.

Back-End-Treiber für ONTAP

Die Back-End-Treiber für ONTAP unterscheiden sich durch das verwendete Protokoll und die Art und Weise, wie die Volumes im Storage-System bereitgestellt werden. Daher sollten Sie bei der Entscheidung, welchen Treiber eingesetzt werden soll, sorgfältig überlegen.

Auf einer höheren Ebene, wenn Ihre Applikation Komponenten hat, die gemeinsamen Storage benötigen (mehrere Pods, die auf dasselbe PVC zugreifen), sind NAS-basierte Treiber die erste Wahl, während die blockbasierten iSCSI-Treiber die Anforderungen von nicht gemeinsam genutztem Storage erfüllen. Wählen Sie das Protokoll basierend auf den Anforderungen der Applikation und der Komfort-Ebene der Storage- und Infrastrukturtteams. Generell besteht für die meisten Applikationen kein Unterschied zwischen ihnen. Oftmals basiert die Entscheidung darauf, ob gemeinsam genutzter Storage (wo mehr als ein POD den gleichzeitigen Zugriff benötigen) benötigt wird.

Die verfügbaren Back-End-Treiber für ONTAP sind:

- `ontap-nas`: Jedes bereitgestellte PV ist ein volles ONTAP FlexVolum.
- `ontap-nas-economy`: Jedes bereitgestellte PV ist ein qtree, mit einer konfigurierbaren Anzahl von qtrees pro FlexVolume (Standard ist 200).
- `ontap-nas-flexgroup`: Jedes PV wird als volle ONTAP FlexGroup bereitgestellt und alle Aggregate werden einer SVM zugewiesen.
- `ontap-san`: Jedes bereitgestellte PV ist eine LUN innerhalb seines eigenen FlexVolume.
- `ontap-san-economy`: Jedes bereitgestellte PV ist eine LUN mit einer konfigurierbaren Anzahl an LUNs pro FlexVolume (Standard ist 100).

Die Auswahl zwischen den drei NAS-Treibern hat einige Auswirkungen auf die Funktionen, die der Applikation zur Verfügung gestellt werden.

Beachten Sie, dass in den nachstehenden Tabellen nicht alle Funktionen durch Astra Trident zugänglich sind. Einige müssen vom Storage-Administrator nach der Bereitstellung angewendet werden, wenn diese Funktion gewünscht wird. Die Super-Skript-Fußnoten unterscheiden die Funktionalität pro Feature und Treiber.

| ONTAP-NAS-Treiber | Snapshot s | Klone | Dynamische Exportrichtlinien | Multi-Anschlus s | QoS | Größe Ändern | Replizierung |
|----------------------------------|----------------|----------------|------------------------------|------------------|----------------|--------------|----------------|
| <code>ontap-nas</code> | Ja. | Ja. | Yes [5] | Ja. | Yes [1] | Ja. | Yes [1] |
| <code>ontap-nas-economy</code> | Jafußnote: 3[] | Jafußnote: 3[] | Yes [5] | Ja. | Jafußnote: 3[] | Ja. | Jafußnote: 3[] |
| <code>ontap-nas-flexgroup</code> | Yes [1] | Nein | Yes [5] | Ja. | Yes [1] | Ja. | Yes [1] |

Astra Trident bietet 2 SAN-Treiber für ONTAP, die unten aufgeführt sind.

| ONTAP SAN-Treiber | Snapshot | Klone | Multi-Anschlus | Bidirektionales CHAP | QoS | Größe Ändern | Replizierung |
|-------------------|----------|-------|----------------|----------------------|----------------|--------------|----------------|
| ontap-san | Ja. | Ja. | Jafußnote: 4[] | Ja. | Yes [1] | Ja. | Yes [1] |
| ontap-san-economy | Ja. | Ja. | Jafußnote: 4[] | Ja. | Jafußnote: 3[] | Ja. | Jafußnote: 3[] |

Fußnote für die obigen Tabellen:

Yes [1]: Wird nicht von Astra Trident gemanagt

Yes [2]: Wird von Astra Trident gemanagt, jedoch nicht mit PV-Granularität

Yes [3]: Wird nicht von Astra Trident gemanagt und nicht von PV-Granularität

Yes [4]: Unterstützt für RAW-Block-Volumes

Yes [5]: Unterstützt von Astra Trident

Die Funktionen, die keine PV-Granularität sind, werden auf das gesamte FlexVolume angewendet, und alle PVs (also qtrees oder LUNs in gemeinsam genutzten FlexVols) teilen einen gemeinsamen Zeitplan.

Wie in den obigen Tabellen zu sehen ist, ist ein Großteil der Funktionalität zwischen den `ontap-nas` Und `ontap-nas-economy` Ist das gleiche. Aber weil die `ontap-nas-economy` Der Fahrer beschränkt die Möglichkeit zur Steuerung des Zeitplans auf PV-Granularität. Dies kann insbesondere Ihre Disaster Recovery- und Backup-Planung beeinträchtigen. Für Entwicklungsteams, die die PVC-Klonfunktion auf ONTAP Storage nutzen möchten, ist dies nur bei Verwendung des möglich `ontap-nas`, `ontap-san` Oder `ontap-san-economy` Treiber.



Der `solidfire-san` Der Treiber ist auch in der Lage, PVCs zu klonen.

Back-End-Treiber für Cloud Volumes ONTAP

Cloud Volumes ONTAP bietet Datenkontrolle und Storage-Funktionen der Enterprise-Klasse für verschiedene Anwendungsfälle, einschließlich Dateifreigaben und Storage-Funktionen auf Blockebene für NAS- und SAN-Protokolle (NFS, SMB/CIFS und iSCSI). Die kompatiblen Treiber für Cloud Volume ONTAP sind `ontap-nas`, `ontap-nas-economy`, `ontap-san` Und `ontap-san-economy`. Diese gelten für Cloud Volume ONTAP für Azure, Cloud Volume ONTAP für GCP.

Back-End-Treiber für Amazon FSX for ONTAP

Amazon FSX for NetApp ONTAP ermöglicht Ihnen die Nutzung von NetApp Funktionen, Performance und Administrationsfunktionen, mit denen Sie vertraut sind, und gleichzeitig die Einfachheit, Agilität, Sicherheit und Skalierbarkeit der Speicherung von Daten auf AWS zu nutzen. FSX für ONTAP unterstützt viele ONTAP-Dateisystemfunktionen und Administrations-APIs. Die kompatiblen Treiber für Cloud Volume ONTAP sind `ontap-nas`, `ontap-nas-economy`, `ontap-nas-flexgroup`, `ontap-san` Und `ontap-san-economy`.

Back-End-Treiber für NetApp HCI/SolidFire

Der `solidfire-san` Der mit den NetApp HCI/SolidFire Plattformen verwendete Treiber unterstützt den Administrator bei der Konfiguration eines Element-Backend für Trident anhand der QoS-Limits. Falls Sie Ihr Backend so entwerfen möchten, dass die spezifischen QoS-Limits für die Volumes gesetzt werden, die durch Trident bereitgestellt werden, verwenden Sie das `type` Parameter in der Backend-Datei. Der Administrator kann auch die Volume-Größe beschränken, die mithilfe von auf dem Storage erstellt werden könnte `limitVolumeSize` Parameter. Momentan werden Element Storage-Funktionen wie die Größenanpassung von Volumes und die Volume-Replizierung von nicht vom unterstützt `solidfire-san` Treiber. Diese Vorgänge sollten manuell über die Web-UI von Element Software durchgeführt werden.

| SolidFire-Treiber | Snapshot s | Klone | Multi-Anschlus s | CHAP | QoS | Größe Ändern | Replizieru ng |
|----------------------------|------------|-------|------------------|------|-----|--------------|---------------|
| <code>solidfire-san</code> | Ja. | Ja. | Jafußnote: 2[] | Ja. | Ja. | Ja. | Yes [1] |

Fußnote:

Yes [1]: Wird nicht von Astra Trident gemanagt

Yes [2]: Unterstützt für RAW-Block-Volumes

Back-End-Treiber für Azure NetApp Files

Astra Trident verwendet den `azure-netapp-files` Treiber für die Verwaltung des "Azure NetApp Dateien" Service:

Weitere Informationen zu diesem Treiber und zur Konfiguration finden Sie unter "[Astra Trident – Back-End-Konfiguration für Azure NetApp Files](#)".

| Azure NetApp Files-Treiber | Snapshots | Klone | Multi-Anschluss | QoS | Erweitern | Replizierung |
|---------------------------------|-----------|-------|-----------------|-----|-----------|--------------|
| <code>azure-netapp-files</code> | Ja. | Ja. | Ja. | Ja. | Ja. | Yes [1] |

Fußnote:

Yes [1]: Wird nicht von Astra Trident gemanagt

Cloud Volumes Service auf Google Cloud Backend-Treiber

Astra Trident verwendet den `gcp-cvs` Treiber zur Verbindung mit der Cloud Volumes Service in der Google Cloud.

Der `gcp-cvs` Der Treiber verwendet virtuelle Pools, um das Backend zu abstrahieren und Astra Trident zu ermöglichen, die Volume-Platzierung zu bestimmen. Der Administrator definiert die virtuellen Pools im `backend.json` Dateien: Storage-Klassen verwenden Selektoren, um virtuelle Pools nach Etikett zu identifizieren.

- Wenn virtuelle Pools im Backend definiert werden, versucht Astra Trident, ein Volume in den Google Cloud Storage-Pools zu erstellen, zu denen diese virtuellen Pools begrenzt sind.
- Wenn virtuelle Pools nicht im Backend definiert sind, wählt Astra Trident aus den verfügbaren Storage-Pools der Region einen Google Cloud Storage-Pool aus.

Um das Google Cloud-Backend auf Astra Trident zu konfigurieren, müssen Sie angeben `projectNumber`, `apiRegion`, und `apiKey` In der Backend-Datei. Die Projektnummer finden Sie in der Google Cloud-Konsole. Der API-Schlüssel wird aus der Datei mit dem privaten Schlüssel des Dienstkontos entnommen, die Sie beim Einrichten des API-Zugriffs für Cloud Volumes Service in der Google Cloud erstellt haben.

Weitere Informationen zu Cloud Volumes Service zu Google Cloud-Servicetypen und Service Levels finden Sie unter "[Erfahren Sie mehr über Astra Trident Support für CVS für GCP](#)".

| Cloud Volumes Service für Google Cloud Treiber | Snapshots | Klone | Multi-Anschluss | QoS | Erweitern | Replizierung |
|--|-----------|-------|-----------------|-----|-----------|--|
| <code>gcp-cvs</code> | Ja. | Ja. | Ja. | Ja. | Ja. | Nur für den CVS-Performance-Diensttyp verfügbar. |



Hinweise zur Replikation

- Replizierung wird nicht durch Astra Trident gemanagt.
- Der Klon wird im selben Speicherpool erstellt wie das Quell-Volume.

Design der Storage-Klasse

Individuelle Storage-Klassen müssen konfiguriert und angewendet werden, um ein Kubernetes Storage Class-Objekt zu erstellen. Dieser Abschnitt erläutert, wie Sie eine Storage-Klasse für Ihre Applikation entwerfen.

Spezifische Back-End-Auslastung

Die Filterung kann innerhalb eines bestimmten Storage-Klassenobjekts verwendet werden, um festzulegen, welcher Storage-Pool bzw. welche Pools für die jeweilige Storage-Klasse verwendet werden sollen. In der Storage-Klasse können drei Filtersätze eingestellt werden: `storagePools`, `additionalStoragePools`, Und/oder `excludeStoragePools`.

Der `storagePools` Parameter hilft bei der Beschränkung des Storage auf Pools, die bestimmten Attributen entsprechen. Der `additionalStoragePools` Mit diesem Parameter wird der Satz von Pools, die Astra Trident zur Bereitstellung verwenden wird, sowie der Reihe von Pools erweitert, die durch die Attribute und ausgewählt wurden `storagePools` Parameter. Sie können entweder nur einen der Parameter oder beide zusammen verwenden, um sicherzustellen, dass der entsprechende Satz von Speicherpools ausgewählt wird.

Der `excludeStoragePools` Parameter wird verwendet, um den aufgelisteten Pool-Satz, der mit den Attributen übereinstimmt, ausdrücklich auszuschließen.

QoS-Richtlinien emulieren

Wenn Sie Storage-Klassen zur Emulation der Quality of Service-Richtlinien entwerfen möchten, erstellen Sie mit dem eine Storage Class `media` Attribut als `hdd` Oder `ssd`. Auf der Grundlage von `media` Attribut, das in der Storage-Klasse erwähnt wird, wählt Trident das entsprechende Back-End aus, das bedient `hdd` Oder `ssd`

Aggregate passen das Medienattribut an und leiten die Bereitstellung der Volumes an das spezifische Aggregat weiter. Deshalb können wir eine Storageklasse PREMIUM schaffen, die hätte `media` Attribut festgelegt als `ssd` Was als PREMIUM-QoS-Richtlinie klassifiziert werden kann. Wir können einen weiteren STANDARD der Storage-Klasse erstellen, bei dem das Medienattribut auf `hdd` gesetzt wäre. Dieser Standard könnte die QoS-Richtlinie SEIN. Darüber hinaus könnten wir das Attribut ```IOPS'` in der Storage-Klasse verwenden, um die Bereitstellung zu einer Element Appliance umzuleiten, die als QoS-Richtlinie definiert werden kann.

Nutzung von Backend basierend auf bestimmten Funktionen

Storage-Klassen ermöglichen die direkte Volume-Bereitstellung an einem bestimmten Back-End, bei dem Funktionen wie Thin Provisioning und Thick Provisioning, Snapshots, Klone und Verschlüsselung aktiviert sind. Um festzulegen, welchen Speicher verwendet werden soll, erstellen Sie Speicherklassen, die das entsprechende Back-End mit aktivierter Funktion angeben.

Virtuelle Pools

Virtuelle Pools sind für alle Astra Trident Back-Ends verfügbar. Sie können virtuelle Pools für jedes Backend mit jedem Treiber von Astra Trident definieren.

Mit virtuellen Pools kann ein Administrator eine Abstraktionsebene über Back-Ends erstellen, auf die über Storage-Klassen verwiesen werden kann. So werden Volumes auf Back-Ends flexibler und effizienter platziert. Verschiedene Back-Ends können mit derselben Serviceklasse definiert werden. Darüber hinaus können mehrere Storage Pools auf demselben Backend erstellt werden, jedoch mit unterschiedlichen Eigenschaften. Wenn eine Storage Class mit einem Selector mit den speziellen Beschriftungen konfiguriert ist, wählt Astra Trident ein Backend, das mit allen Auswahlketten übereinstimmt, um das Volume zu platzieren. Wenn die Storage Class Selector mit mehreren Storage Pools übereinstimmt, wählt Astra Trident einen von ihnen für die Bereitstellung des Volume aus.

Virtual Pool Design

Beim Erstellen eines Backend können Sie im Allgemeinen eine Reihe von Parametern angeben. Der Administrator konnte kein weiteres Back-End mit denselben Storage Credentials und anderen Parametern erstellen. Mit der Einführung von virtuellen Pools wurde dieses Problem behoben. Virtual Pools ist eine Ebene-Abstraktion, die zwischen dem Backend und der Kubernetes Storage Class eingeführt wird. So kann der Administrator Parameter zusammen mit Labels definieren, die über Kubernetes Storage Klassen als Selektion auf Backend-unabhängige Weise referenziert werden können. Virtuelle Pools können mit Astra Trident für alle unterstützten NetApp Back-Ends definiert werden. Dazu zählen SolidFire/NetApp HCI, ONTAP, Cloud Volumes Service auf GCP und Azure NetApp Files.



Bei der Definition von virtuellen Pools wird empfohlen, nicht zu versuchen, die Reihenfolge vorhandener virtueller Pools in einer Backend-Definition neu anzuordnen. Es wird auch empfohlen, Attribute für einen vorhandenen virtuellen Pool nicht zu bearbeiten/zu ändern und stattdessen einen neuen virtuellen Pool zu definieren.

Emulation verschiedener Service-Level/QoS

Es ist möglich, virtuelle Pools zur Emulation von Serviceklassen zu entwerfen. Untersuchen wir mit der Implementierung des virtuellen Pools für den Cloud Volume Service für Azure NetApp Files, wie wir verschiedene Serviceklassen einrichten können. Konfigurieren Sie das Azure NetApp Files Back-End mit mehreren Labels, die unterschiedliche Performance-Levels repräsentieren. Einstellen `servicelevel` Dem entsprechenden Leistungslevel hinzuzufügen und unter jeder Beschriftung weitere erforderliche Aspekte hinzuzufügen. Erstellen Sie nun verschiedene Kubernetes Storage-Klassen, die verschiedenen virtuellen Pools zugeordnet werden würden. Verwenden der `parameters.selector` Feld, jede StorageClass ruft auf, welche

virtuellen Pools zum Hosten eines Volumes verwendet werden dürfen.

Zuweisen eines spezifischen Satzes von Aspekten

Mehrere virtuelle Pools mit spezifischen Aspekten können über ein einzelnes Storage-Back-End entwickelt werden. Konfigurieren Sie dazu das Backend mit mehreren Beschriftungen und legen Sie die erforderlichen Aspekte unter jedem Etikett fest. Erstellen Sie jetzt mit dem verschiedene Kubernetes-Storage-Klassen `parameters.selector` Feld, das verschiedenen virtuellen Pools zugeordnet werden würde. Die Volumes, die im Backend bereitgestellt werden, werden im ausgewählten virtuellen Pool über die Aspekte definiert.

PVC-Merkmale, die die Storage-Bereitstellung beeinflussen

Einige Parameter außerhalb der angeforderten Storage-Klasse können sich bei der Erstellung eines PVC auf den Entscheidungsprozess von Astra Trident auswirken.

Zugriffsmodus

Wenn Sie Speicher über ein PVC anfordern, ist eines der Pflichtfelder der Zugriffsmodus. Der gewünschte Modus kann sich auf das ausgewählte Backend auswirken, um die Speicheranforderung zu hosten.

Astra Trident versucht, das verwendete Storage-Protokoll mit der in der folgenden Matrix angegebenen Zugriffsmethode abzustimmen. Dies ist unabhängig von der zugrunde liegenden Storage-Plattform.

| | ReadWriteOnce | ReadOnlyManche | ReadWriteViele |
|-------|----------------------|-----------------------|-----------------------|
| ISCSI | Ja. | Ja. | Ja (Raw Block) |
| NFS | Ja. | Ja. | Ja. |

Eine Anfrage nach einem ReadWriteManche PVC, die an eine Trident-Implementierung ohne konfiguriertes NFS-Backend gesendet werden, führt dazu, dass kein Volume bereitgestellt wird. Aus diesem Grund sollte der Anforderer den Zugriffsmodus verwenden, der für seine Anwendung geeignet ist.

Volume-Vorgänge

Persistente Volumes ändern

Persistente Volumes sind mit zwei Ausnahmen unveränderliche Objekte in Kubernetes. Sobald die Rückgewinnungsrichtlinie erstellt wurde, kann die Größe geändert werden. Dies hindert jedoch nicht daran, einige Aspekte des Volumes außerhalb von Kubernetes zu ändern. Das kann durchaus wünschenswert sein, wenn das Volume für spezifische Applikationen angepasst werden soll, um sicherzustellen, dass die Kapazität nicht versehentlich verbraucht wird oder das Volume einfach aus irgendeinem Grund auf einen anderen Storage Controller verschoben werden kann.



Kubernetes-in-Tree-Provisioners unterstützen derzeit keine Vorgänge zur Größenanpassung von Volumes für NFS oder iSCSI PVS. Astra Trident unterstützt die Erweiterung von NFS- und iSCSI-Volumes.

Die Verbindungsdetails des PV können nach der Erstellung nicht geändert werden.

Erstellung von On-Demand-Volume-Snapshots

Astra Trident unterstützt die On-Demand-Volume-Snapshot-Erstellung und die Erstellung von PVCs aus Snapshots mithilfe des CSI-Frameworks. Snapshots bieten eine bequeme Methode, zeitpunktgenaue Kopien

der Daten zu erstellen und haben unabhängig vom Quell-PV in Kubernetes einen Lebenszyklus. Diese Snapshots können zum Klonen von PVCs verwendet werden.

Volumes-Erstellung aus Snapshots

Astra Trident unterstützt außerdem die Erstellung von PersistenzVolumes aus Volume Snapshots. Um dies zu erreichen, erstellen Sie einfach ein PersistenzVolumeClaim und erwähnen die `datasource` Als den benötigten Snapshot, aus dem das Volume erstellt werden muss. Astra Trident wird dieses PVC behandeln, indem ein Volume mit den auf dem Snapshot vorhandenen Daten erstellt wird. Mit dieser Funktion können Daten regionsübergreifend dupliziert, Testumgebungen erstellt, ein defektes oder defektes Produktionsvolumen vollständig ersetzt oder bestimmte Dateien und Verzeichnisse abgerufen und auf ein anderes angeschlossenes Volume übertragen werden.

Verschieben Sie Volumes im Cluster

Storage-Administratoren können Volumes zwischen Aggregaten und Controllern im ONTAP Cluster unterbrechungsfrei für den Storage-Nutzer verschieben. Dieser Vorgang wirkt sich nicht auf Astra Trident oder den Kubernetes-Cluster aus, solange das Zielaggregat eine der SVM ist, auf die Astra Trident Zugriff hat. Was noch wichtiger ist: Wenn das Aggregat neu zur SVM hinzugefügt wurde, muss das Backend durch erneutes Hinzufügen zu Astra Trident aktualisiert werden. Dies führt Astra Trident dazu, die SVM neu zu inventarisieren, damit das neue Aggregat erkannt wird.

Das Verschieben von Volumes zwischen Back-Ends wird von Astra Trident jedoch nicht automatisch unterstützt. Dazu gehören SVMs im selben Cluster, zwischen Clustern oder auf einer anderen Storage-Plattform (auch wenn dieses Storage-System mit Astra Trident verbunden ist).

Wenn ein Volume an einen anderen Speicherort kopiert wird, kann die Funktion zum Importieren aktueller Volumes in Astra Trident verwendet werden.

Erweitern Sie Volumes

Astra Trident unterstützt die Anpassung von NFS und iSCSI PVS. Dadurch können Benutzer ihre Volumes direkt über die Kubernetes-Ebene skalieren. Eine Volume-Erweiterung ist für alle größeren NetApp Storage-Plattformen möglich, einschließlich ONTAP, SolidFire/NetApp HCI und Cloud Volumes Service Back-Ends. Um eine mögliche Erweiterung später zu ermöglichen, stellen Sie fest `allowVolumeExpansion` Bis `true` In Ihrer StorageClass, die mit dem Volume verbunden ist. Wenn die Größe des Persistent Volume geändert werden muss, bearbeiten Sie den `spec.resources.requests.storage` Anmerkung im Persistent Volume Claim zur erforderlichen Volume-Größe. Trident übernimmt automatisch die Anpassung der Größe des Volumes im Storage-Cluster.

Importieren eines vorhandenen Volumes in Kubernetes

Mit dem Volume-Import kann ein vorhandenes Storage Volume in eine Kubernetes-Umgebung importiert werden. Dies wird derzeit von unterstützt `ontap-nas`, `ontap-nas-flexgroup`, `solidfire-san`, `azure-netapp-files`, und `gcp-cvs` Treiber. Diese Funktion ist hilfreich, wenn Sie eine vorhandene Applikation in Kubernetes oder während Disaster-Recovery-Szenarien portieren.

Bei Verwendung von ONTAP und `solidfire-san` Treiber, verwenden Sie den Befehl `tridentctl import volume <backend-name> <volume-name> -f /path/pvc.yaml` Um ein vorhandenes Volume in Kubernetes zu importieren, das von Astra Trident gemanagt werden soll Die im Befehl „Importvolumen“ verwendete PVC-YAML- oder JSON-Datei weist auf eine Storage-Klasse hin, die Astra Trident als bereitstellung identifiziert. Stellen Sie bei Verwendung eines NetApp HCI/SolidFire Backend sicher, dass die Volume-Namen eindeutig sind. Wenn die Volume-Namen dupliziert sind, klonen Sie das Volume auf einen eindeutigen Namen, sodass die Funktion zum Importieren des Volumes zwischen diesen Namen

unterscheiden kann.

Wenn der `azure-netapp-files` Oder `gcp-cvs` Treiber wird verwendet, verwenden Sie den Befehl `tridentctl import volume <backend-name> <volume path> -f /path/pvc.yaml` Um das Volume in Kubernetes zu importieren, das von Astra Trident gemanagt werden soll. Dadurch wird eine eindeutige Volumenreferenz sichergestellt.

Wenn der obige Befehl ausgeführt wird, wird Astra Trident das Volume auf dem Backend finden und seine Größe lesen. Die Volume-Größe der konfigurierten PVC wird automatisch hinzugefügt (und bei Bedarf überschrieben). Astra Trident erstellt dann das neue PV und Kubernetes bindet die PVC an das PV.

Wenn ein Container so eingesetzt wurde, dass er das spezifische importierte PVC benötigt, bleibt er in einem ausstehenden Zustand, bis das PVC/PV-Paar über den Volumenimport gebunden ist. Nachdem das PVC/PV-Paar gebunden ist, sollte der Behälter aufstehen, sofern keine anderen Probleme auftreten.

OpenShift Services implementieren

Die Cluster-Services OpenShift mit großem Mehrwert bieten Clusteradministratoren und den gehosteten Applikationen wichtige Funktionen. Der Storage, den diese Services nutzen, kann mithilfe der Node-lokalen Ressourcen bereitgestellt werden. Dadurch wird jedoch häufig die Kapazität, Performance, Wiederherstellbarkeit und die Nachhaltigkeit des Service begrenzt. Die Nutzung eines Enterprise-Speicher-Arrays zur Bereitstellung der Kapazität für diese Services kann einen erheblich verbesserten Service ermöglichen. OpenShift und die Speicheradministratoren sollten jedoch eng zusammenarbeiten, um die besten Optionen für die einzelnen zu bestimmen. Die Red hat-Dokumentation sollte intensiv genutzt werden, um die Anforderungen zu ermitteln und sicherzustellen, dass die Anforderungen hinsichtlich Größe und Leistung erfüllt werden.

Registry-Service

Der Einsatz und das Management von Storage für die Registrierung wurde am dokumentiert "[netapp.io](#)" Im "[Blog](#)".

Protokollierungsservice

Wie andere OpenShift-Services wird auch der Protokollierungsservice mithilfe von Ansible mit Konfigurationsparametern bereitgestellt, die von der Bestandsdatei auch bekannt sind Hosts, die im Playbook zur Verfügung gestellt werden. Es gibt zwei Installationsmethoden, die behandelt werden: Bereitstellung der Protokollierung während der ersten OpenShift-Installation und Bereitstellung der Protokollierung nach OpenShift Installiert.



Ab Red hat OpenShift Version 3.9 empfiehlt die offizielle Dokumentation gegen NFS für den Protokollierungsservice, da sie Bedenken hinsichtlich Datenbeschädigung hat. Dies basiert auf Red hat Tests ihrer Produkte. Der ONTAP NFS-Server weist diese Probleme nicht auf und kann problemlos eine Protokollierungsbereitstellung zurücksichern. Letztendlich liegt die Wahl des Protokolls für den Protokollierungsservice bei Ihnen. Ich weiß nur, dass beide bei der Nutzung von NetApp Plattformen hervorragend funktionieren. Es gibt keinen Grund, NFS zu vermeiden, wenn dies Ihre Präferenz ist.

Wenn Sie sich für die Verwendung von NFS mit dem Protokollierungsservice entscheiden, müssen Sie die Ansible-Variable festlegen `openshift_enable_unsupported_configurations` Bis `true` Um zu verhindern, dass der Installer ausfällt.

Los geht's

Der Protokollierungsservice kann optional sowohl für Applikationen als auch für die Kernvorgänge des OpenShift-Clusters selbst implementiert werden. Wenn Sie sich für die Bereitstellung der Betriebsprotokollierung entscheiden, geben Sie die Variable `openshift_logging_use_ops` Als `true`, Zwei Instanzen des Dienstes werden erstellt. Die Variablen, die die Protokollierungsinstanz für Vorgänge steuern, enthalten darin "OPS", während die Instanz für Anwendungen nicht.

Das Konfigurieren der Ansible-Variablen gemäß der Implementierungsmethode ist wichtig, um sicherzustellen, dass der richtige Storage von den zugrunde liegenden Services verwendet wird. Betrachten wir nun die Optionen für die einzelnen Bereitstellungsmethoden.



Die folgenden Tabellen enthalten nur die für die Speicherkonfiguration relevanten Variablen, die sich auf den Protokollierungsservice beziehen. Weitere Optionen finden Sie in "[Logging-Dokumentation von redhat OpenShift](#)". Die entsprechend Ihrer Bereitstellung überprüft, konfiguriert und verwendet werden sollten.

Die Variablen in der folgenden Tabelle führen dazu, dass im Ansible-Playbook ein PV und eine PVC für den Protokollierungsservice erstellt werden. Diese Details werden verwendet. Diese Methode ist wesentlich weniger flexibel als nach der Installation von OpenShift das Playbook für die Komponenteninstallation zu verwenden. Wenn Sie jedoch vorhandene Volumes zur Verfügung haben, ist dies eine Option.

| Variabel | Details |
|--|---|
| <code>openshift_logging_storage_kind</code> | Auf einstellen <code>nfs</code> So erstellen Sie ein NFS-PV für den Protokollierungsservice. |
| <code>openshift_logging_storage_host</code> | Der Hostname oder die IP-Adresse des NFS-Hosts. Diese Einstellung sollte auf die Daten-LIF für Ihre Virtual Machine eingestellt sein. |
| <code>openshift_logging_storage_nfs_directory</code> | Der Mount-Pfad für den NFS-Export. Beispiel: Wenn das Volume mit verbunden ist <code>/openshift_logging</code> , Sie würden diesen Pfad für diese Variable verwenden. |
| <code>openshift_logging_storage_volume_name</code> | Der Name, z.B. <code>pv_ose_logs</code> , Des zu erstellenden PV. |
| <code>openshift_logging_storage_volume_size</code> | Beispielsweise die Größe des NFS-Exports <code>100Gi</code> . |

Wenn Ihr OpenShift-Cluster bereits ausgeführt wird und daher Trident implementiert und konfiguriert wurde, kann das Installationsprogramm die Volumes mithilfe der dynamischen Provisionierung erstellen. Die folgenden Variablen müssen konfiguriert werden.

| Variabel | Details |
|--|---|
| <code>openshift_logging_es_pvc_dynamic</code> | Setzen Sie auf „true“, um dynamisch bereitgestellte Volumes zu verwenden. |
| <code>openshift_logging_es_pvc_storage_class_name</code> | Der Name der Speicherklasse, die in der PVC verwendet wird. |
| <code>openshift_logging_es_pvc_size</code> | Die Größe des im PVC angeforderten Volumens. |
| <code>openshift_logging_es_pvc_prefix</code> | Ein Präfix für die vom Protokollierungsservice verwendeten VES. |

| Variabel | Details |
|--|---|
| <code>openshift_logging_es_ops_pvc_dynamic</code> | Auf einstellen <code>true</code> Um dynamisch bereitgestellte Volumes für die OPS-Protokollierungsinstanz zu verwenden. |
| <code>openshift_logging_es_ops_pvc_storage_class_name</code> | Der Name der Speicherklasse für die OPS-Protokollierungsinstanz. |
| <code>openshift_logging_es_ops_pvc_size</code> | Die Größe der Volume-Anforderung für die OPS-Instanz. |
| <code>openshift_logging_es_ops_pvc_prefix</code> | Ein Präfix für die OPS-Instanz VES. |

Bereitstellen des Protokollierungs-Stacks

Wenn Sie die Protokollierung als Teil des ursprünglichen OpenShift-Installationsprozesses bereitstellen, müssen Sie nur den Standardprozess für die Bereitstellung befolgen. Ansible konfiguriert und implementiert die erforderlichen Services und OpenShift-Objekte, sodass der Service sobald Ansible abgeschlossen ist.

Wenn Sie die Implementierung jedoch nach der Erstinstallation durchführen, muss das Komponenten-Playbook von Ansible verwendet werden. Dieser Prozess kann sich mit verschiedenen Versionen von OpenShift leicht ändern, also lesen und folgen ["Dokumentation der redhat OpenShift Container Platform 3.11"](#) Für Ihre Version.

Kennzahlungsservice

Der Kennzahlungsservice liefert dem Administrator wertvolle Informationen zum Status, zur Ressourcenauslastung und zur Verfügbarkeit des OpenShift-Clusters. Dies ist zudem für die automatische Pod-Funktionalität erforderlich, und viele Unternehmen nutzen die Daten des Kennzahlungsservice für ihre Kostenabrechnung und/oder die Anzeige von Applikationen.

Wie beim Protokollierungsservice und OpenShift als Ganzes wird auch Ansible für die Implementierung des Kennzahlungsservice verwendet. Ebenso wie der Protokollierungsservice kann der Metrikservice während der ersten Einrichtung des Clusters oder nach dessen Betrieb mithilfe der Installationsmethode für Komponenten bereitgestellt werden. Die folgenden Tabellen enthalten die Variablen, die für die Konfiguration von persistentem Storage für den Kennzahlungsservice wichtig sind.



Die nachfolgenden Tabellen enthalten nur die Variablen, die für die Storage-Konfiguration relevant sind, da sie sich auf den Kennzahlenservice beziehen. Es gibt viele andere Optionen in der Dokumentation gefunden, die entsprechend Ihrer Bereitstellung überprüft, konfiguriert und verwendet werden sollten.

| Variabel | Details |
|---|---|
| <code>openshift_metrics_storage_kind</code> | Auf einstellen <code>nfs</code> So erstellen Sie ein NFS-PV für den Protokollierungsservice. |
| <code>openshift_metrics_storage_host</code> | Der Hostname oder die IP-Adresse des NFS-Hosts. Diese Einstellung sollte auf die Daten-LIF für Ihre SVM eingestellt sein. |

| Variabel | Details |
|--|---|
| <code>openshift_metrics_storage_nfs_directory</code> | Der Mount-Pfad für den NFS-Export. Beispiel: Wenn das Volume mit verbunden ist <code>/openshift_metrics</code> , Sie würden diesen Pfad für diese Variable verwenden. |
| <code>openshift_metrics_storage_volume_name</code> | Der Name, Z. B. <code>pv_ose_metrics</code> , Des zu erstellenden PV. |
| <code>openshift_metrics_storage_volume_size</code> | Beispielsweise die Größe des NFS-Exports 100Gi. |

Wenn Ihr OpenShift-Cluster bereits ausgeführt wird und daher Trident implementiert und konfiguriert wurde, kann das Installationsprogramm die Volumes mithilfe der dynamischen Provisionierung erstellen. Die folgenden Variablen müssen konfiguriert werden.

| Variabel | Details |
|---|---|
| <code>openshift_metrics_cassandra_pvc_prefix</code> | Ein Präfix, das für die PVCs der Kennzahlen verwendet wird. |
| <code>openshift_metrics_cassandra_pvc_size</code> | Die Größe der Volumes, die angefordert werden sollen. |
| <code>openshift_metrics_cassandra_storage_type</code> | Der Storage-Typ, der für Metriken verwendet werden soll. Dieser muss für Ansible auf dynamisch festgelegt sein, um PVCs mit der entsprechenden Storage-Klasse zu erstellen. |
| <code>openshift_metrics_cassandra_pvc_storage_class_name</code> | Der Name der zu verwendenden Speicherklasse. |

Bereitstellen des Kennzahlenservice

Implementieren Sie den Service mithilfe von Ansible, wenn Sie die entsprechenden Ansible-Variablen in der Host-/Inventardatei festlegen. Wenn Sie zur Installationszeit OpenShift bereitstellen, wird das PV automatisch erstellt und verwendet. Wenn Sie mit den Komponenten-Playbooks implementieren, erstellt Ansible nach der Installation von OpenShift alle erforderlichen PVCs. Nachdem Astra Trident Storage für sie bereitgestellt hat, kann der Service implementiert werden.

Die oben genannten Variablen und der Prozess für die Bereitstellung können sich mit jeder Version von OpenShift ändern. Überprüfen und befolgen Sie die Anweisungen ["Der OpenShift-Implementierungslaufplan von Red hat"](#) Für Ihre Version so konfigurieren, dass sie für Ihre Umgebung konfiguriert ist.

Datensicherung und Disaster Recovery

Informieren Sie sich über die Sicherungs- und Recovery-Optionen für Astra Trident und Volumes, die mit Astra Trident erstellt wurden. Für jede Applikation mit einer Persistenzanforderung sollte eine Datensicherungs- und Recovery-Strategie eingesetzt werden.

Astra Trident Replizierung und Recovery

Sie können ein Backup erstellen, um Astra Trident im Falle eines Ausfalls wiederherzustellen.

Astra Trident Replizierung

Astra Trident verwendet Kubernetes CRDs zum Speichern und Managen seines eigenen Zustands sowie des Kubernetes-Clusters und etcd zum Speichern seiner Metadaten.

Schritte

1. Sichern Sie den Kubernetes-Cluster und den Einsatz von "[Kubernetes: Backup eines uscd-Clusters](#)".
2. Platzieren Sie die Backup-Artefakte auf einer FlexVol.



Wir empfehlen, die SVM, auf der sich die FlexVol befindet, mit einer SnapMirror-Beziehung zu einer anderen SVM zu sichern.

Astra Trident Recovery

Mit Kubernetes CRDs und dem Kubernetes-Cluster uscd Snapshot können Sie Astra Trident wiederherstellen.

Schritte

1. Mounten Sie von der Ziel-SVM das Volume, das die Kubernetes usw.-Datendateien und Zertifikate enthält, auf dem Host, der als Master-Node eingerichtet wird.
2. Kopieren Sie alle erforderlichen Zertifikate zum Kubernetes-Cluster unter `/etc/kubernetes/pki` Und die etcd-Mitgliedsdateien unter `/var/lib/etcd`.
3. Stellen Sie das Kubernetes-Cluster aus dem etcd-Backup mit wieder her "[Kubernetes: Wiederherstellung eines uscd-Clusters](#)".
4. Laufen `kubectl get crd` Um zu überprüfen, ob alle benutzerdefinierten Trident Ressourcen eingerichtet sind, und rufen Sie die Trident Objekte ab, um zu überprüfen, ob alle Daten verfügbar sind.

SVM-Replizierung und Recovery

Astra Trident kann keine Replizierungsbeziehungen konfigurieren. Storage-Administrator kann diese jedoch verwenden "[ONTAP SnapMirror](#)" Um eine SVM zu replizieren.

Bei einem Notfall können Sie die SnapMirror Ziel-SVM aktivieren, um die Datenbereitstellung zu starten. Sie können zurück zum primären System wechseln, wenn die Systeme wiederhergestellt sind.

Über diese Aufgabe

Bei Verwendung der SnapMirror SVM-Replizierungsfunktion sind die folgenden Überlegungen zu beachten:

- Sie sollten für jede SVM ein eigene Back-End mit aktivierter SVM-DR erstellen.
- Konfigurieren Sie die Storage-Klassen so, dass die replizierten Back-Ends nur bei Bedarf ausgewählt werden, um zu vermeiden, dass Volumes ohne Replizierung auf den Back-Ends bereitgestellt werden, die SVM-DR unterstützen.
- Applikationsadministratoren sollten sich über die zusätzlichen Kosten und die Komplexität der Replizierung informieren und ihren Recovery-Plan vor Beginn des Prozesses sorgfältig prüfen.

SVM-Replizierung

Verwenden Sie können "[ONTAP: SnapMirror SVM-Replizierung](#)" Um die SVM-Replikationsbeziehung zu erstellen.

Mit SnapMirror können Sie festlegen, was repliziert werden soll. Sie müssen wissen, welche Optionen Sie

beim Preforming ausgewählt haben [SVM-Recovery mit Astra Trident](#).

- ["-Identität-bewahren wahr"](#) Replizierung der gesamten SVM-Konfiguration
- ["-Discard-configs Netzwerk"](#) Davon sind LIFs und zugehörige Netzwerkeinstellungen nicht enthalten.
- ["-Identity-preserve false"](#) Repliziert nur die Volumes und die Sicherheitskonfiguration.

SVM-Recovery mit Astra Trident

Astra Trident erkennt SVM-Ausfälle nicht automatisch. Bei einem Notfall kann der Administrator das Trident Failover manuell auf die neue SVM initialisieren.

Schritte

1. Abbrechen geplanter und laufender SnapMirror Übertragungen, Abbrechen der Replizierungsbeziehung, stoppen Sie die Quell-SVM und aktivieren Sie dann die SnapMirror Ziel-SVM.
2. Wenn Sie angegeben haben `-identity-preserve false` Oder `-discard-config network` Aktualisieren Sie beim Konfigurieren der SVM-Replikation die `managementLIF` Und `dataLIF` In der Trident Back-End-Definitionsdatei.
3. Bestätigen `storagePrefix` Ist in der Definitionsdatei des Trident-Backends vorhanden. Dieser Parameter kann nicht geändert werden. Auslassung `storagePrefix` Führt dazu, dass das Backend-Update fehlschlägt.
4. Aktualisieren Sie alle erforderlichen Back-Ends, um den neuen Ziel-SVM-Namen widerzuspiegeln. Verwenden Sie dazu Folgendes:

```
./tridentctl update backend <backend-name> -f <backend-json-file> -n  
<namespace>
```

5. Wenn Sie angegeben haben `-identity-preserve false` Oder `discard-config network`, Sie müssen alle Anwendungen Pods hüpfen.



Wenn Sie angegeben haben `-identity-preserve true` Alle von Astra Trident bereitgestellten Volumes beginnen, Daten bereitzustellen, wenn die Ziel-SVM aktiviert ist.

Volume-Replizierung und Recovery

Astra Trident kann keine SnapMirror Replizierungsbeziehungen konfigurieren. Der Storage-Administrator kann diese jedoch verwenden ["Replizierung und Recovery mit ONTAP SnapMirror"](#) Um von Astra Trident erstellte Volumes zu replizieren.

Sie können dann die wiederhergestellten Volumes in Astra Trident mit importieren ["Tridentctl-Volumenimport"](#).



Import wird auf nicht unterstützt `ontap-nas-economy`, `ontap-san-economy`, Oder `ontap-flexgroup-economy` Treiber.

Snapshot Datensicherung

Sie können Daten schützen und wiederherstellen mit:

- Ein externer Snapshot-Controller und CRDs zum Erstellen von Kubernetes-Volume-Snapshots von persistenten Volumes (PVs).

["Volume Snapshots"](#)

- ONTAP Snapshots zur Wiederherstellung der gesamten Inhalte eines Volumes oder zur Wiederherstellung einzelner Dateien oder LUNs.

["ONTAP Snapshots"](#)

Applikationsreplizierung für Astra Control Center

Mithilfe von Astra Control können Sie Daten und Applikationsänderungen mithilfe von asynchronen Replizierungsfunktionen von SnapMirror von einem Cluster zu einem anderen replizieren.

["Astra Control: Replizierung von Applikationen auf ein Remote-System mithilfe von SnapMirror Technologie"](#)

Sicherheit

Sicherheit

Stellen Sie mit den hier aufgeführten Empfehlungen sicher, dass Ihre Astra Trident Installation sicher ist.

Führen Sie Astra Trident in einem eigenen Namespace aus

Es ist wichtig, dass Applikationen, Applikationsadministratoren, Benutzer und Managementapplikationen auf die Objektdefinitionen von Astra Trident oder die Pods zugreifen können, um zuverlässigen Storage sicherzustellen und potenzielle schädliche Aktivitäten zu blockieren.

Zur Trennung der anderen Applikationen und Benutzer von Astra Trident muss immer Astra Trident in einem eigenen Kubernetes Namespace installiert werden (`trident`). Wenn Astra Trident in einem eigenen Namespace bereitgestellt wird, wird sichergestellt, dass nur die Administratoren von Kubernetes auf den Astra Trident Pod und die Artefakte (z. B. Backend und CHAP-Schlüssel, falls zutreffend) zugreifen können, die in den namenweisen CRD-Objekten gespeichert sind.

Sie sollten sicherstellen, dass nur Administratoren Zugriff auf den Astra Trident Namespace und damit auf das `tridentctl` Applikation haben.

Verwenden Sie CHAP-Authentifizierung mit ONTAP SAN Back-Ends

Astra Trident unterstützt die CHAP-basierte Authentifizierung für ONTAP-SAN-Workloads (mithilfe von `ontap-san` und `ontap-san-economy` Treiber). NetApp empfiehlt die Verwendung von bidirektionalem CHAP mit Astra Trident zur Authentifizierung zwischen einem Host und dem Storage-Backend.

Bei ONTAP-Back-Ends, die die SAN-Storage-Treiber verwenden, kann Astra Trident bidirektionales CHAP einrichten und CHAP-Benutzernamen und -Schlüssel über `manage tridentctl`.

Siehe ["Hier"](#) Um zu erfahren, wie Astra Trident CHAP auf ONTAP Back-Ends konfiguriert.

Verwenden Sie CHAP-Authentifizierung mit NetApp HCI und SolidFire Back-Ends

NetApp empfiehlt die Implementierung von bidirektionalem CHAP, um die Authentifizierung zwischen einem Host und den NetApp HCI und SolidFire Back-Ends zu gewährleisten. Astra Trident verwendet ein geheimes

Objekt mit zwei CHAP-Passwörtern pro Mandant. Wenn Astra Trident installiert ist, managt es die CHAP-Schlüssel und speichert sie in einem `tridentvolume` CR-Objekt für das jeweilige PV. Bei der Erstellung eines PV verwendet Astra Trident die CHAP-Schlüssel, um eine iSCSI-Sitzung zu initiieren und mit dem NetApp HCI- und dem SolidFire-System über CHAP zu kommunizieren.



Die von Astra Trident erstellten Volumes sind keiner Volume Access Group zugeordnet.

Nutzen Sie Astra Trident mit NVE und NAE

NetApp ONTAP bietet Verschlüsselung ruhender Daten zum Schutz sensibler Daten, wenn eine Festplatte gestohlen, zurückgegeben oder einer neuen Verwendung zugewiesen wird. Weitere Informationen finden Sie unter "[NetApp Volume Encryption Übersicht konfigurieren](#)".

- Wenn NAE auf dem Backend aktiviert ist, wird jedes im Astra Trident bereitgestellte Volume NAE-aktiviert.
- Wenn NAE im Backend nicht aktiviert ist, wird jedes in Astra Trident bereitgestellte Volume mit NVE aktiviert, es sei denn, Sie setzen das NVE-Verschlüsselungsflag auf `false`. Bei der Back-End-Konfiguration:

Volumes, die in Astra Trident auf einem NAE-fähigen Back-End erstellt werden, müssen NVE oder NAE-verschlüsselt sein.



- Sie können das NVE-Verschlüsselungsflag auf `true` einstellen. In der Trident-Back-End-Konfiguration können Sie die NAE-Verschlüsselung außer Kraft setzen und für jedes Volume einen bestimmten Verschlüsselungsschlüssel verwenden.
- Setzen des NVE-Verschlüsselungsflag auf `false`. Auf einem NAE-fähigen Back-End wird ein NAE-fähiges Volume erstellt. Sie können die NAE-Verschlüsselung nicht deaktivieren, indem Sie das NVE-Verschlüsselungsflag auf `false` setzen.

- Sie können in Astra Trident manuell ein NVE-Volume erstellen, indem Sie explizit das NVE-Verschlüsselungsflag auf `true` festlegen.

Weitere Informationen zu Back-End-Konfigurationsoptionen finden Sie unter:

- "[ONTAP SAN-Konfigurationsoptionen](#)"
- "[NAS-Konfigurationsoptionen von ONTAP](#)"

Linux Unified Key Setup (LUKS)

Sie können Linux Unified Key Setup (LUKS) aktivieren, um ONTAP SAN und ONTAP SAN ECONOMY Volumes auf Astra Trident zu verschlüsseln. Astra Trident unterstützt die Rotation von Passphrase und die Volume-Erweiterung für LUKS-verschlüsselte Volumes.

In Astra Trident verwenden LUKS-verschlüsselte Volumes den `aes-xts-plain64` Zypher und den Modus, wie von empfohlen "[NIST](#)".

Bevor Sie beginnen

- Worker Nodes müssen `cryptsetup` 2.1 oder höher (aber unter 3.0) installiert sein. Weitere Informationen finden Sie unter "[Gitlab: Cryptsetup](#)".
- Aus Performance-Gründen wird empfohlen, dass Arbeiterknoten Advanced Encryption Standard New

Instructions (AES-NI) unterstützen. Führen Sie den folgenden Befehl aus, um die Unterstützung von AES-NI zu überprüfen:

```
grep "aes" /proc/cpuinfo
```

Wenn nichts zurückgegeben wird, unterstützt Ihr Prozessor nicht AES-NI. Weitere Informationen zu AES-NI finden Sie unter: ["Intel: Advanced Encryption Standard Instructions \(AES-NI\)"](#).

Aktivieren Sie die LUKS-Verschlüsselung

Sie können die Verschlüsselung auf Host-Seite pro Volume mithilfe von Linux Unified Key Setup (LUKS) für ONTAP SAN und ONTAP SAN ECONOMY Volumes aktivieren.

Schritte

1. Definieren Sie LUKS-Verschlüsselungsattribute in der Backend-Konfiguration. Weitere Informationen zu den Back-End-Konfigurationsoptionen für ONTAP SAN finden Sie unter ["ONTAP SAN-Konfigurationsoptionen"](#).

```
"storage": [  
  {  
    "labels":{"luks": "true"},  
    "zone":"us_east_1a",  
    "defaults": {  
      "luksEncryption": "true"  
    }  
  },  
  {  
    "labels":{"luks": "false"},  
    "zone":"us_east_1a",  
    "defaults": {  
      "luksEncryption": "false"  
    }  
  },  
]
```

2. Nutzung `parameters.selector` So definieren Sie die Speicherpools mit LUKS-Verschlüsselung. Beispiel:

```

apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: luks
provisioner: netapp.io/trident
parameters:
  selector: "luks=true"
  csi.storage.k8s.io/node-stage-secret-name: luks-${pvc.name}
  csi.storage.k8s.io/node-stage-secret-namespace: ${pvc.namespace}

```

3. Erstellen Sie ein Geheimnis, das die LUKS-Passphrase enthält. Beispiel:

```

kubectl -n trident create -f luks-pvc1.yaml
apiVersion: v1
kind: Secret
metadata:
  name: luks-pvc1
stringData:
  luks-passphrase-name: A
  luks-passphrase: secretA

```

Einschränkungen

LUKS-verschlüsselte Volumes können die ONTAP Deduplizierung und Komprimierung nicht nutzen.

Back-End-Konfiguration zum Importieren von LUKS-Volumes

Um ein LUKS-Volume zu importieren, müssen Sie festlegen `luksEncryption` Bis(`true` Am Backend. Der `luksEncryption` Die Option teilt Astra Trident mit, ob das Volume LUKS-konform ist (`true`) Oder nicht LUKS-konform (`false`) Wie im folgenden Beispiel gezeigt.

```

version: 1
storageDriverName: ontap-san
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
svm: trident_svm
username: admin
password: password
defaults:
  luksEncryption: 'true'
  spaceAllocation: 'false'
  snapshotPolicy: default
  snapshotReserve: '10'

```

Eine LUKS-Passphrase drehen

Sie können die LUKS-Passphrase drehen und die Drehung bestätigen.



Vergessen Sie keine Passphrase, bis Sie überprüft haben, dass sie nicht mehr von einem Volume, einem Snapshot oder einem geheimen Schlüssel referenziert wird. Wenn eine referenzierte Passphrase verloren geht, können Sie das Volume möglicherweise nicht mounten und die Daten bleiben verschlüsselt und unzugänglich.

Über diese Aufgabe

DIE Drehung der LUKS-Passphrase erfolgt, wenn ein Pod, das das Volume bindet, nach der Angabe einer neuen LUKS-Passphrase erstellt wird. Bei der Erstellung eines neuen Pods vergleicht Astra Trident die LUKS-Passphrase auf dem Volume mit der aktiven Passphrase im Geheimnis.

- Wenn die Passphrase auf dem Volume nicht mit der aktiven Passphrase im Geheimnis übereinstimmt, erfolgt die Drehung.
- Wenn die Passphrase auf dem Volume mit der aktiven Passphrase im Geheimnis übereinstimmt, wird das angezeigte `previous-luks-passphrase` Parameter wird ignoriert.

Schritte

1. Fügen Sie die hinzu `node-publish-secret-name` Und `node-publish-secret-namespace` StorageClass-Parameter. Beispiel:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: csi-san
provisioner: csi.trident.netapp.io
parameters:
  trident.netapp.io/backendType: "ontap-san"
  csi.storage.k8s.io/node-stage-secret-name: luks
  csi.storage.k8s.io/node-stage-secret-namespace: ${pvc.namespace}
  csi.storage.k8s.io/node-publish-secret-name: luks
  csi.storage.k8s.io/node-publish-secret-namespace: ${pvc.namespace}
```

2. Identifizieren Sie vorhandene Passphrases auf dem Volume oder Snapshot.

Datenmenge

```
tridentctl -d get volume luks-pvc1
GET http://127.0.0.1:8000/trident/v1/volume/<volumeID>

...luksPassphraseNames: ["A"]
```

Snapshot

```
tridentctl -d get snapshot luks-pvc1
GET http://127.0.0.1:8000/trident/v1/volume/<volumeID>/<snapshotID>

...luksPassphraseNames: ["A"]
```

3. Aktualisieren Sie das LUKS-Geheimnis für das Volume, um die neuen und vorherigen Passphrases anzugeben. Unbedingt `previous-luke-passphrase-name` Und `previous-luks-passphrase` Übereinstimmung mit der vorherigen Passphrase.

```
apiVersion: v1
kind: Secret
metadata:
  name: luks-pvc1
stringData:
  luks-passphrase-name: B
  luks-passphrase: secretB
  previous-luks-passphrase-name: A
  previous-luks-passphrase: secretA
```

4. Erstellen Sie einen neuen Pod, der das Volume montiert. Dies ist erforderlich, um die Rotation zu initiieren.
5. Überprüfen Sie, ob die Passphrase gedreht wurde.

Datenmenge

```
tridentctl -d get volume luks-pvc1
GET http://127.0.0.1:8000/trident/v1/volume/<volumeID>

...luksPassphraseNames: ["B"]
```

Snapshot

```
tridentctl -d get snapshot luks-pvc1
GET http://127.0.0.1:8000/trident/v1/volume/<volumeID>/<snapshotID>

...luksPassphraseNames: ["B"]
```

Ergebnisse

Die Passphrase wurde gedreht, wenn nur die neue Passphrase auf dem Volume und dem Snapshot zurückgegeben wird.



Werden beispielsweise zwei Passphrases zurückgegeben `luksPassphraseNames: ["B", "A"]`, Die Rotation ist unvollständig. Sie können einen neuen Pod auslösen, um zu versuchen, die Rotation abzuschließen.

Aktivieren Sie die Volume-Erweiterung

Sie können Volume-Erweiterung auf einem LUKS-verschlüsselten Volume aktivieren.

Schritte

1. Aktivieren Sie die `CSINodeExpandSecret` Funktionstor (Beta 1.25+). Siehe ["Kubernetes 1.25: Verwenden Sie Secrets zur Node-gesteuerten Erweiterung von CSI Volumes"](#) Entsprechende Details.
2. Fügen Sie die hinzu `node-expand-secret-name` Und `node-expand-secret-namespace` `StorageClass`-Parameter. Beispiel:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: luks
provisioner: netapp.io/trident
parameters:
  selector: "luks=true"
  csi.storage.k8s.io/node-stage-secret-name: luks-{{pvc.name}}
  csi.storage.k8s.io/node-stage-secret-namespace: {{pvc.namespace}}
  csi.storage.k8s.io/node-expand-secret-name: luks-{{pvc.name}}
  csi.storage.k8s.io/node-expand-secret-namespace: {{pvc.namespace}}
allowVolumeExpansion: true
```

Ergebnisse

Wenn Sie die Online-Speichererweiterung initiieren, gibt das Kubelet die entsprechenden Zugangsdaten an den Treiber weiter.

Wissen und Support

Häufig gestellte Fragen

Hier finden Sie Antworten auf die häufig gestellten Fragen zur Installation, Konfiguration, Aktualisierung und Fehlerbehebung von Astra Trident.

Allgemeine Fragen

Wie oft wird Astra Trident veröffentlicht?

Astra Trident wird alle drei Monate veröffentlicht: Januar, April, Juli und Oktober. Dies ist ein Monat nach der Kubernetes-Version.

Unterstützt Astra Trident alle Funktionen, die in einer bestimmten Version von Kubernetes verfügbar sind?

Astra Trident unterstützt in der Regel keine Alpha-Funktionen in Kubernetes. Trident unterstützt möglicherweise Beta-Funktionen in den beiden Trident Versionen, die nach der Kubernetes Beta-Version folgen.

Verfügt Astra Trident über irgendwelche Abhängigkeiten von anderen NetApp Produkten für seine Funktionsweise?

Astra Trident ist unabhängig von anderen NetApp Softwareprodukten und kann als eigenständige Applikation eingesetzt werden. Sie sollten jedoch ein NetApp Back-End Storage-Gerät haben.

Wie erhalte ich vollständige Astra Trident Konfigurationsdetails?

Verwenden Sie die `tridentctl get` Befehl für weitere Informationen über Ihre Astra Trident Konfiguration.

Kann ich Metriken abrufen, wie Storage von Astra Trident bereitgestellt wird?

Ja. Prometheus Endpunkte, die zur Erfassung von Informationen über den Astra Trident Vorgang verwendet werden können, z. B. die Anzahl der gemanagten Back-Ends, die Anzahl der bereitgestellten Volumes, die verbrauchten Byte usw. Sie können auch verwenden "[Einblicke in die Cloud](#)" Zur Überwachung und Analyse.

Ändert sich die Benutzererfahrung, wenn Astra Trident als CSI-Bereitstellung verwendet wird?

Nein Es gibt keine Änderungen hinsichtlich der Benutzerfreundlichkeit und Funktionalitäten. Der bereitstellungsname wird verwendet `csi.trident.netapp.io`. Diese Methode zur Installation von Astra Trident ist empfehlenswert, wenn Sie alle neuen Funktionen der aktuellen und zukünftigen Versionen nutzen möchten.

Installation und Verwendung von Astra Trident in einem Kubernetes Cluster

Unterstützt Astra Trident eine Offline-Installation von einer privaten Registry?

Ja, Astra Trident kann offline installiert werden. Siehe "[Hier](#)".

Kann Astra Trident Remote installiert werden?

Ja. Astra Trident 18.10 und höher unterstützen Remote-Installationsfunktionen von jedem Rechner aus `kubectl` Zugriff auf das Cluster. Nachher `kubectl` Der Zugriff wird verifiziert (z. B. initiieren Sie ein `kubectl get nodes` Befehl vom Remotegerät zur Überprüfung), folgen Sie den Installationsanweisungen.

Kann ich Hochverfügbarkeit mit Astra Trident konfigurieren?

Astra Trident ist als Kubernetes Deployment (ReplicaSet) mit einer Instanz installiert und ist daher mit integrierter HA ausgestattet. Sie sollten die Anzahl der Replikat in der Bereitstellung nicht erhöhen. Wenn der Node, auf dem Astra Trident installiert ist, verloren geht oder der POD nicht mehr zur Verfügung steht, implementiert Kubernetes den Pod automatisch wieder zu einem funktionierenden Node im Cluster. Astra Trident ist nur auf der Kontrollebene, sodass aktuell montierte Pods nicht beeinträchtigt werden, wenn Astra Trident neu implementiert wird.

Benötigt Astra Trident Zugriff auf den kube-System-Namespace?

Astra Trident liest den Kubernetes API Server aus, um zu bestimmen, wann Applikationen neue PVCs anfordern. Daher ist der Zugriff auf das kube-System erforderlich.

Welche Rollen und Privilegien werden von Astra Trident verwendet?

Das Trident-Installationsprogramm erstellt ein Kubernetes ClusterRole, das spezifischen Zugriff auf die Ressourcen PersistentVolume, PersistentVolumeClaim, StorageClass und Secret des Kubernetes-Clusters hat. Siehe "[Hier](#)".

Kann ich lokal die genauen Manifest-Dateien generieren, die Astra Trident zur Installation verwendet?

Sie können die genauen Manifest-Dateien, die Astra Trident für die Installation verwendet, lokal generieren und ändern, falls erforderlich. Siehe "[Hier](#)".

Kann ich dieselbe ONTAP Backend-SVM für zwei separate Astra Trident Instanzen für zwei separate Kubernetes Cluster nutzen?

Obwohl dies nicht empfohlen wird, können Sie für zwei Astra Trident Instanzen dieselbe Backend-SVM verwenden. Geben Sie während der Installation einen eindeutigen Volume-Namen für jede Instanz an und/oder geben Sie einen eindeutigen Namen an `StoragePrefix` Parameter in `setup/backend.json` Datei: Dadurch wird sichergestellt, dass nicht dieselbe FlexVol für beide Instanzen verwendet wird.

Ist es möglich, Astra Trident unter ContainerLinux (früher CoreOS) zu installieren?

Astra Trident ist einfach ein Kubernetes Pod und kann überall installiert werden, wo Kubernetes ausgeführt wird.

Kann ich Astra Trident mit NetApp Cloud Volumes ONTAP verwenden?

Ja, Astra Trident wird unterstützt auf AWS, Google Cloud und Azure.

Funktioniert Astra Trident mit Cloud Volumes Services?

Ja, Astra Trident unterstützt den Azure NetApp Files-Service in Azure und die Cloud Volumes Service in GCP.

Fehlerbehebung und Support

Bietet NetApp Unterstützung für Astra Trident?

Auch wenn Astra Trident kostenlos über Open-Source-Software bereitgestellt wird, unterstützt NetApp das System vollständig, vorausgesetzt, Ihr NetApp Backend wird unterstützt.

Wie kann ich einen Support-Fall anheben?

Wenn Sie einen Support-Case anheben möchten, führen Sie einen der folgenden Schritte aus:

1. Kontaktieren Sie Ihren Support Account Manager und erhalten Sie Hilfe bei der Ticketausstellung.
2. Eröffnen Sie einen Support-Case, indem Sie Kontakt aufnehmen ["NetApp Support"](#).

Wie generiere ich ein Support Log-Paket?

Sie können ein Support-Bundle erstellen, indem Sie ausführen `tridentctl logs -a`. Erfassen Sie zusätzlich zu den im Bundle erfassten Protokollen das kubelet-Protokoll, um die Mount-Probleme auf der Seite von Kubernetes zu diagnostizieren. Die Anweisungen zum Abrufen des kubelet-Protokolls variieren je nach der Installation von Kubernetes.

Was muss ich tun, wenn ich einen Antrag auf eine neue Funktion stellen muss?

Erstellen Sie ein Problem bei ["Astra Trident Github"](#) Und erwähnen Sie **RFE** im Thema und Beschreibung der Ausgabe.

Wo kann ich einen Defekt aufwerfen?

Erstellen Sie ein Problem bei ["Astra Trident Github"](#). Achten Sie darauf, alle erforderlichen Informationen und Protokolle für das Problem einzubeziehen.

Was passiert, wenn ich schnell Fragen zu Astra Trident habe, die ich klären muss? Gibt es eine Gemeinschaft oder ein Forum?

Wenn Sie Fragen, Probleme oder Wünsche haben, wenden Sie sich über unseren Astra an uns ["Kanal abstecken"](#) Oder GitHub.

Das Passwort meines Storage-Systems hat sich geändert und Astra Trident funktioniert nicht mehr. Wie kann ich das Recovery durchführen?

Aktualisieren Sie das Back-End-Passwort mit `tridentctl update backend myBackend -f </path/to_new_backend.json> -n trident.Austausch myBackend` Im Beispiel mit Ihrem Backend-Namen, und ``/path/to_new_backend.json` Mit dem Pfad zum richtigen `backend.json` Datei:

Astra Trident kann meinen Kubernetes-Node nicht finden. Wie kann ich das beheben?

Es gibt zwei wahrscheinliche Szenarien, warum Astra Trident keinen Kubernetes-Node finden kann. Dies kann auf ein Netzwerkproblem innerhalb von Kubernetes oder auf ein DNS-Problem zurückzuführen sein. Das Trident Node-Demonset, das auf jedem Kubernetes Node ausgeführt wird, muss mit dem Trident Controller kommunizieren können, um den Node bei Trident zu registrieren. Wenn nach der Installation von Astra Trident Netzwerkänderungen aufgetreten sind, treten dieses Problem nur mit den neuen Kubernetes-Nodes auf, die dem Cluster hinzugefügt werden.

Geht der Trident Pod verloren, gehen die Daten verloren?

Daten gehen nicht verloren, wenn der Trident Pod zerstört wird. Trident Metadaten werden in CRD-Objekten gespeichert. Alle PVS, die von Trident bereitgestellt wurden, funktionieren ordnungsgemäß.

Upgrade Astra Trident

Kann ich ein Upgrade von einer älteren Version direkt auf eine neuere Version durchführen (einige Versionen werden übersprungen)?

NetApp unterstützt das Upgrade des Astra Trident von einer Hauptversion auf das nächste sofort größere Release. Sie können ein Upgrade von Version 18.xx auf 19.xx, 19.xx auf 20.xx usw. durchführen. Sie sollten das Upgrade vor der Implementierung in einer Produktionsumgebung in einem Labor testen.

Ist es möglich, Trident auf eine vorherige Version herunterzustufen?

Wenn Sie eine Korrektur für Fehler benötigen, die nach einem Upgrade, Abhängigkeitsproblemen oder einem nicht erfolgreichen oder unvollständigen Upgrade beobachtet wurden, sollten Sie dies tun ["Deinstallieren Sie Astra Trident"](#) Und installieren Sie die frühere Version mithilfe der entsprechenden Anweisungen für diese Version neu. Dies ist der einzige empfohlene Weg, um ein Downgrade auf eine frühere Version.

Back-Ends und Volumes managen

Muss ich Management- und Daten-LIFs in einer ONTAP-Back-End-Definitionsdatei definieren?

Die Management-LIF ist erforderlich. Logische Datenschnittstelle variiert:

- ONTAP SAN: Nicht für iSCSI angeben. Astra Trident verwendet ["ONTAP selektive LUN-Zuordnung"](#) Um die iSCSI LIFs zu ermitteln, die für die Einrichtung einer Multi-Path-Sitzung erforderlich sind. Wenn eine Warnung erzeugt wird `dataLIF` ist explizit definiert. Siehe ["ONTAP SAN-Konfigurationsoptionen und -Beispiele"](#) Entsprechende Details.
- ONTAP-NAS: Wir empfehlen, anzugeben `dataLIF`. Falls nicht vorgesehen, ruft Astra Trident Daten-LIFs von der SVM ab. Sie können einen vollständig qualifizierten Domänennamen (FQDN) angeben, der für die NFS-Mount-Vorgänge verwendet werden soll. Damit können Sie ein Round-Robin-DNS zum Load-Balancing über mehrere Daten-LIFs erstellen. Siehe ["ONTAP-NAS-Konfigurationsoptionen und Beispiele"](#) Entsprechende Details

Kann Astra Trident CHAP für ONTAP-Back-Ends konfigurieren?

Ja. Astra Trident unterstützt bidirektionales CHAP für ONTAP Back-Ends. Dazu ist eine Einstellung erforderlich `useCHAP=true` Der Back-End-Konfiguration durchgeführt.

Wie schaffe ich Exportrichtlinien mit Astra Trident?

Astra Trident kann Exportrichtlinien ab Version 20.04 dynamisch erstellen und verwalten. Dadurch kann der Storage-Administrator einen oder mehrere CIDR-Blöcke in seiner Back-End-Konfiguration bereitstellen und Trident Add-Node-IPs erstellen, die einer erstellten Exportrichtlinie innerhalb dieses Bereichs liegen. Auf diese Weise managt Astra Trident das Hinzufügen und Löschen von Regeln für Knoten mit IPs innerhalb der angegebenen CIDRs automatisch.

Können IPv6-Adressen für das Management und die Daten-LIFs verwendet werden?

Astra Trident unterstützt die Definition von IPv6-Adressen für:

- `managementLIF` Und `dataLIF` Für ONTAP-NAS-Back-Ends.
- `managementLIF` Für ONTAP SAN Back-Ends. Sie können nicht angeben `dataLIF` Auf einem ONTAP-SAN-Back-End

Astra Trident muss über das Flag installiert werden `--use-ipv6` (Für `tridentctl` Installation), `IPv6` (Für Trident Operator), oder `tridentTPv6` (Für Helm-Installation), damit die IT über IPv6 funktioniert.

Ist es möglich, die Management LIF auf dem Backend zu aktualisieren?

Ja, es ist möglich, die Backend-Management-LIF mithilfe des zu aktualisieren `tridentctl update backend` Befehl.

Ist es möglich, die Daten-LIF auf dem Backend zu aktualisieren?

Sie können die Daten-LIF auf aktualisieren `ontap-nas` Und `ontap-nas-economy` Nur.

Kann ich in Astra Trident mehrere Back-Ends für Kubernetes erstellen?

Astra Trident kann viele Back-Ends gleichzeitig unterstützen, entweder mit demselben oder mit unterschiedlichen Treibern.

Wie speichert Astra Trident Back-End-Anmeldedaten?

Astra Trident speichert die Backend-Anmeldedaten als Kubernetes Secrets.

Wie wählt Astra Trident ein spezifisches Backend aus?

Wenn die Back-End-Attribute nicht zur automatischen Auswahl der richtigen Pools für eine Klasse verwendet werden können, wird das verwendet `storagePools` Und `additionalStoragePools` Parameter werden zur Auswahl eines bestimmten Pools verwendet.

Wie kann ich sicherstellen, dass Astra Trident nicht über ein spezifisches Backend bereitgestellt wird?

Der `excludeStoragePools` Parameter wird verwendet, um den Pool-Satz, den Astra Trident zur Bereitstellung verwenden wird, zu filtern und alle Pools, die übereinstimmen, zu entfernen.

Wenn es mehrere Back-Ends derselben Art gibt, wie wählt Astra Trident das zu verwendende Back-End aus?

Wenn es mehrere konfigurierte Back-Ends desselben Typs gibt, wählt Astra Trident basierend auf den in vorhandenen Parametern das entsprechende Backend aus `StorageClass` Und `PersistentVolumeClaim`. Wenn es beispielsweise mehrere `ontap-nas`-Treiber-Back-Ends gibt, versucht Astra Trident, die Parameter im zu entsprechen `StorageClass` Und `PersistentVolumeClaim` Kombinieren Sie ein Backend, das die in aufgeführten Anforderungen erfüllen kann `StorageClass` Und `PersistentVolumeClaim`. Wenn die Anfrage mit mehreren Back-Ends übereinstimmt, wählt Astra Trident aus einem dieser Back-Ends nach dem Zufallsprinzip aus.

Unterstützt Astra Trident bidirektionales CHAP mit Element/SolidFire?

Ja.

Wie implementiert Astra Trident qtrees auf einem ONTAP Volume? Wie viele qtrees können auf einem einzelnen Volume implementiert werden?

Der `ontap-nas-economy` Der Treiber erstellt bis zu 200 qtrees in derselben FlexVol (konfigurierbar zwischen 50 und 300), 100,000 qtrees pro Cluster Node und 2,4 Mio. pro Cluster. Wenn Sie eine neue eingeben `PersistentVolumeClaim` Das wird vom Wirtschaftstreiber gewartet und der Fahrer sieht danach aus, ob es bereits eine FlexVol gibt, die den neuen Qtree bedienen kann. Wenn es keine FlexVol gibt, die für den Qtree Services bereitstellen können, wird eine neue FlexVol erstellt.

Wie kann ich Unix Berechtigungen für Volumes festlegen, die auf ONTAP NAS bereitgestellt werden?

Sie können Unix-Berechtigungen auf dem von Astra Trident bereitgestellten Volume festlegen, indem Sie einen Parameter in der Backend-Definitionsdatei festlegen.

Wie kann ich bei der Bereitstellung eines Volumes einen expliziten Satz von ONTAP-NFS-Mount-Optionen konfigurieren?

Standardmäßig stellt Astra Trident keine Mount-Optionen für Kubernetes auf jeden Wert ein. Befolgen Sie das angegebene Beispiel, um die Mount-Optionen in der Kubernetes Storage-Klasse anzugeben "[Hier](#)".

Wie lege ich die bereitgestellten Volumes auf eine bestimmte Exportrichtlinie fest?

Um den entsprechenden Hosts den Zugriff auf ein Volume zu erlauben, verwenden Sie das `exportPolicy` In der Backend-Definitionsdatei konfigurierter Parameter.

Wie setze ich mit ONTAP die Volume-Verschlüsselung durch Astra Trident ein?

Sie können die Verschlüsselung auf dem von Trident bereitgestellten Volume mit dem Verschlüsselungsparameter in der Back-End-Definitionsdatei festlegen. Weitere Informationen finden Sie unter: "[Astra Trident arbeitet mit NVE und NAE zusammen](#)"

Wie implementiert man QoS für ONTAP am besten über Astra Trident?

Nutzung `StorageClasses` Bei der Implementierung von QoS für ONTAP.

Wie soll ich über Astra Trident Thin oder Thick Provisioning angeben?

Die ONTAP-Treiber unterstützen entweder Thin Provisioning oder Thick Provisioning. Die ONTAP-Treiber verwenden Thin Provisioning standardmäßig. Wenn Thick Provisioning gewünscht ist, sollten Sie entweder die Back-End-Definitionsdatei oder die konfigurieren `StorageClass`. Wenn beide konfiguriert sind, `StorageClass` Hat Vorrang. Konfigurieren Sie Folgendes für ONTAP:

1. Ein `StorageClass`, Einstellen Sie die `provisioningType` Attribut als dick.
2. Aktivieren Sie in der Back-End-Definitionsdatei die Option `Thick Volumes backend spaceReserve parameter` Als Volumen.

Wie kann ich sicherstellen, dass die verwendeten Volumes nicht gelöscht werden, auch wenn ich aus Versehen die PVC lösche?

Der PVC-Schutz ist für Kubernetes ab Version 1.10 automatisch aktiviert.

Kann ich die von Astra Trident erstellten NFS PVCs ausbauen?

Ja. Sie können ein von Astra Trident erstelltes PVC erweitern. Beachten Sie, dass Volume Autogrow eine ONTAP-Funktion ist, die nicht für Trident geeignet ist.

Kann ich ein Volume importieren, während es sich in SnapMirror Data Protection (DP) oder offline Modus befindet?

Der Volumenimport schlägt fehl, wenn sich das externe Volume im DP-Modus befindet oder offline ist. Sie erhalten die folgende Fehlermeldung:

```
Error: could not import volume: volume import failed to get size of
volume: volume <name> was not found (400 Bad Request) command terminated
with exit code 1.
Make sure to remove the DP mode or put the volume online before importing
the volume.
```

Wie wird ein Ressourcenkontingent auf ein NetApp Cluster übersetzt?

Die Kubernetes-Storage-Ressourcen-Quota sollte so lange funktionieren, wie NetApp Storage die Kapazität hat. Wenn der NetApp Storage die Kubernetes-Kontingenteinstellungen aus Mangel an Kapazität nicht erfüllen kann, versucht Astra Trident, die Bereitstellung zu übernehmen, aber Fehler zu beheben.

Kann ich mit Astra Trident Volume Snapshots erstellen?

Ja. Der Einsatz von On-Demand-Volume-Snapshots und persistenten Volumes aus Snapshots wird von Astra Trident unterstützt. Um PVS aus Snapshots zu erstellen, stellen Sie sicher, dass das `VolumeSnapshotDataSource` Feature Gate ist aktiviert.

Welche Faktoren sind die Faktoren, die die Volume-Snapshots von Astra Trident unterstützen?

Ab heute ist die Unterstützung von On-Demand Snapshot für unser verfügbar `ontap-nas`, `ontap-nas-flexgroup`, `ontap-san`, `ontap-san-economy`, `solidfire-san`, `gcp-cvs`, und `azure-netapp-files` Back-End-Treiber:

Wie kann ich ein Snapshot-Backup eines von Astra Trident bereitgestellten Volumes mit ONTAP erstellen?

Dies ist auf verfügbar `ontap-nas`, `ontap-san`, und `ontap-nas-flexgroup` Treiber. Sie können auch ein angeben `snapshotPolicy` Für das `ontap-san-economy` Treiber auf FlexVol-Ebene.

Dies ist auch auf der verfügbar `ontap-nas-economy` Treiber, aber auf der FlexVol-Ebene-Granularität und nicht auf der qtree-Ebene Granularität. Damit die von Astra Trident bereitgestellte Snapshot-Volumes unterstützt werden können, legen Sie die Back-End-Parameter-Option fest `snapshotPolicy` Zu der gewünschten Snapshot-Policy, wie im ONTAP-Back-End definiert. Alle Snapshots, die vom Storage Controller gemacht werden, sind durch Astra Trident nicht bekannt.

Kann ich einen prozentualen Anteil der Snapshot-Reserve für ein über Astra Trident bereitgestelltes Volume festlegen?

Ja, Sie können einen bestimmten Prozentsatz des Speicherplatzes zum Speichern der Snapshot-Kopien durch Astra Trident reservieren, indem Sie den einstellen `snapshotReserve` Attribut in der Back-End-

Definitionsdatei. Wenn Sie konfiguriert haben `snapshotPolicy` Und `snapshotReserve` In der Back-End-Definitionsdatei wird der Prozentsatz der Snapshot-Reserve entsprechend gesetzt `snapshotReserve` In der Back-End-Datei erwähnten Prozentsatz. Wenn der `snapshotReserve` Prozentzahl ist nicht erwähnt, ONTAP nimmt standardmäßig den Prozentsatz der Snapshot-Reserve als 5 an. Wenn der `snapshotPolicy` Option ist auf „none“ gesetzt, der Prozentsatz der Snapshot-Reserve ist auf 0 gesetzt.

Kann ich direkt auf das Snapshot-Verzeichnis des Volumes zugreifen und Dateien kopieren?

Ja, Sie können auf das Snapshot-Verzeichnis auf dem von Trident bereitgestellten Volume zugreifen, indem Sie das festlegen `snapshotDir` Parameter in der Backend-Definitionsdatei.

Kann ich SnapMirror für Volumes über Astra Trident einrichten?

Derzeit muss SnapMirror extern über ONTAP CLI oder OnCommand System Manager festgelegt werden.

Wie kann ich persistente Volumes auf einen bestimmten ONTAP Snapshot wiederherstellen?

So stellen Sie ein Volume auf einem ONTAP-Snapshot wieder her:

1. Legen Sie den Applikations-POD still, der das persistente Volume nutzt.
2. Zurücksetzen des erforderlichen Snapshots mithilfe von ONTAP CLI oder OnCommand System Manager
3. Starten Sie den Anwendungs-POD neu.

Kann Trident Volumes auf SVMs bereitstellen, die ein Load Sharing Mirror konfiguriert haben?

Load-Sharing-Spiegelungen können für Root-Volumes von SVMs erstellt werden, die Daten über NFS bereitstellen. ONTAP aktualisiert automatisch die Spiegelungen zur Lastverteilung für Volumes, die von Trident erstellt wurden. Dies kann zu Verzögerungen bei der Montage der Volumes führen. Wenn mehrere Volumes mit Trident erstellt werden, hängt die Bereitstellung eines Volumes davon ab, ob ONTAP die Load-Sharing-Spiegelung aktualisiert.

Wie lässt sich die Storage-Klassennutzung für jeden Kunden/Mandanten trennen?

Kubernetes erlaubt Storage-Klassen nicht in Namespaces. Kubernetes lässt sich jedoch mithilfe von Storage-Ressourcenkontingenten, die pro Namespace gelten, die Nutzung einer bestimmten Storage-Klasse pro Namespace begrenzen. Um einem bestimmten Namespace-Zugriff auf einen bestimmten Speicher zu verweigern, setzen Sie das Ressourcenkontingent für diese Speicherklasse auf 0.

Fehlerbehebung

Verwenden Sie die hier angegebenen Hinweise zur Fehlerbehebung bei Problemen, die bei der Installation und Verwendung von Astra Trident möglicherweise auftreten können.

Allgemeine Fehlerbehebung

- Falls der Trident Pod nicht richtig angezeigt wird (z. B. wenn er im nicht mehr ordnungsgemäß funktioniert ContainerCreating Phase mit weniger als zwei einsatzbereiten Containern), Laufen `kubectl -n trident describe deployment trident` Und `kubectl -n trident describe pod trident--**` Dieser Service ermöglicht Ihnen Einblick. Abrufen von Kubelet-Protokollen (z. B. über `journalctl -xeu kubelet`) Kann auch hilfreich sein.
- Wenn die Informationen in den Trident-Protokollen nicht genügend sind, können Sie versuchen, den

Debug-Modus für Trident zu aktivieren, indem Sie den übergeben `-d` Markieren Sie anhand Ihrer Installationsoption den Installationsparameter.

Bestätigen Sie dann, dass Debug mit eingestellt ist `./tridentctl logs -n trident` Und suchen nach `level=debug msg` Im Protokoll.

Mit Operator installiert

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

Dadurch werden alle Trident Pods neu gestartet, was mehrere Sekunden dauern kann. Sie können dies überprüfen, indem Sie die Spalte „ALTER“ in der Ausgabe von beobachten `kubectl get pod -n trident`.

Für den Einsatz von Astra Trident 20.07 und 20.10 `tprov` Anstelle von `torc`.

Installiert mit Helm

```
helm upgrade <name> trident-operator-21.07.1-custom.tgz --set
tridentDebug=true`
```

Mit tridentctl installiert

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

- Sie können auch Debug-Protokolle für jedes Backend erhalten, indem Sie eingeschlossen `debugTraceFlags` Back-End-Definition: Beispiel `debugTraceFlags: {"api":true, "method":true, }` Zum Abrufen von API-Aufrufen und Methodentraversierungen in den Trident-Protokollen. Vorhandene Back-Ends können eine `debugTraceFlags` Konfiguriert mit einem `tridentctl backend update`.
- Stellen Sie bei der Verwendung von RedHat CoreOS sicher, dass `iscsid` Ist auf den Worker-Knoten aktiviert und standardmäßig gestartet. Dies kann mit OpenShift MachineConfigs oder durch Ändern der Zündvorlagen erfolgen.
- Ein häufiges Problem kann bei der Verwendung von Trident mit auftreten "[Azure NetApp Dateien](#)" Wenn die Mandanten- und Client-Geheimnisse von einer App-Registrierung mit unzureichenden Berechtigungen stammen. Eine vollständige Liste der Trident-Anforderungen finden Sie unter "[Azure NetApp Dateien](#)" Konfiguration.
- Bei Problemen mit der Montage eines PV in einem Behälter, darauf achten `rpcbind` Wird installiert und ausgeführt. Verwenden Sie den erforderlichen Paket-Manager für das Host-Betriebssystem, und überprüfen Sie, ob `rpcbind` Wird ausgeführt. Sie können den Status des überprüfen `rpcbind` Service durch Ausführen eines `systemctl status rpcbind` Oder gleichwertige Informationen.
- Wenn ein Trident Back-End meldet, dass es sich im befindet `failed` Status, obwohl er zuvor gearbeitet hat, wird wahrscheinlich dadurch verursacht, dass die mit dem Backend verbundenen SVM/Admin-Berechtigungen geändert werden. Aktualisieren der Back-End-Informationen mit `tridentctl update backend` Oder wenn Sie auf den Trident Pod verzichten, wird dieses Problem behoben.

- Wenn bei der Installation von Trident mit Docker als Container-Laufzeit Probleme mit Berechtigungen auftreten, versuchen Sie die Installation von Trident mit dem `--in cluster=false` Flagge. Dadurch wird kein Installateur-Pod verwendet und es werden keine Berechtigungs-Probleme vermieden, die aufgrund des angezeigt werden `trident-installer` Benutzer:
- Verwenden Sie die `uninstall` parameter `<Uninstalling Trident>` Zum Reinigen nach einem fehlgeschlagenen Lauf. Standardmäßig werden die von Trident erstellten CRDs nicht vom Skript entfernt, sodass es sicher ist, auch in einer laufenden Implementierung zu deinstallieren und wieder zu installieren.
- Wenn Sie ein Downgrade auf eine frühere Version von Trident durchführen möchten, führen Sie zuerst die aus `tridentctl uninstall` Befehl zum Entfernen von Trident. Laden Sie die gewünschten herunter "[Trident Version](#)" Und installieren Sie mit `tridentctl install` Befehl.
- Nach erfolgreicher Installation, wenn ein PVC in der stecken bleibt `Pending Phase`, Ausführen `kubectl describe pvc` Kann zusätzliche Informationen darüber angeben, warum Trident ein PV für diese PVC nicht bereitgestellt hat.

Fehlerbehebung bei einer nicht erfolgreichen Trident-Implementierung mithilfe des Betreibers

Wenn Sie Trident über den Operator implementieren, lautet der 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, indem Sie folgenden Befehl ausführen:

```
tridentctl logs -l trident-operator
```

Das Nachführen der Protokolle des Dreizack-Operators kann auf den Punkt verweisen, an dem das Problem liegt. Ein solches Problem könnte beispielsweise darin liegen, dass die erforderlichen Container-Images nicht von vorgelagerten Registern in einer Airgoed-Umgebung übertragen werden können.

Um zu verstehen, warum die Installation von Trident nicht erfolgreich war, you Sehen Sie sich die an `TridentOrchestrator` Status:

```

kubect1 describe torc trident-2
Name:          trident-2
Namespace:
Labels:        <none>
Annotations:   <none>
API Version:   trident.netapp.io/v1
Kind:          TridentOrchestrator
...
Status:
  Current Installation Params:
    IPv6:
    Autosupport Hostname:
    Autosupport Image:
    Autosupport Proxy:
    Autosupport Serial Number:
    Debug:
    Image Pull Secrets:          <nil>
    Image Registry:
    k8sTimeout:
    Kubelet Dir:
    Log Format:
    Silence Autosupport:
    Trident Image:
  Message:          Trident is bound to another CR 'trident'
  Namespace:        trident-2
  Status:           Error
  Version:
Events:
  Type          Reason  Age                From                Message
  ----          -
  Warning       Error   16s (x2 over 16s)  trident-operator.netapp.io  Trident
is bound to another CR 'trident'

```

Dieser Fehler weist darauf hin, dass bereits ein vorhanden ist TridentOrchestrator
Darüber wurde Trident installiert. Da jeder Kubernetes-Cluster nur kann
Haben Sie eine Instanz von Trident, der Operator sorgt dafür, dass auf jeden Fall
Zeit gibt es nur einen aktiven TridentOrchestrator Das kann er
Erstellen.

Zusätzlich können Sie durch die Beobachtung des Status der Trident Pods oft angeben, ob etwas nicht richtig
ist.

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

| NAME | READY | STATUS | RESTARTS |
|--|-------|------------------|----------|
| trident-csi-4p5kq 5m18s | 1/2 | ImagePullBackOff | 0 |
| trident-csi-6f45bfd8b6-vfrkw 5m19s | 4/5 | ImagePullBackOff | 0 |
| trident-csi-9q5xc 5m18s | 1/2 | ImagePullBackOff | 0 |
| trident-csi-9v95z 5m18s | 1/2 | ImagePullBackOff | 0 |
| trident-operator-766f7b8658-ldzsv 8m17s | 1/1 | Running | 0 |

Sie sehen deutlich, dass die Pods nicht vollständig initialisiert werden können
Weil ein oder mehrere Container-Bilder nicht abgerufen wurden.

Um das Problem zu beheben, sollten Sie die bearbeiten `TridentOrchestrator` CR.
Alternativ können Sie auch löschen `TridentOrchestrator` Und erstellen Sie eine neue
Eine mit der geänderten und genauen Definition.

Fehlerbehebung bei einer nicht erfolgreichen Trident-Implementierung mit `tridentctl`

Um herauszufinden, was schief gelaufen ist, können Sie den Installer mit dem `renew` Argument, das den Debug-Modus aktiviert und Ihnen hilft zu verstehen, was das Problem ist:

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

Nachdem Sie das Problem behoben haben, können Sie die Installation wie folgt bereinigen und dann den `tridentctl install` Befehl erneut:

```
./tridentctl uninstall -n trident  
INFO Deleted Trident deployment.  
INFO Deleted cluster role binding.  
INFO Deleted cluster role.  
INFO Deleted service account.  
INFO Removed Trident user from security context constraint.  
INFO Trident uninstallation succeeded.
```

Entfernen Sie Astra Trident und CRDs vollständig

Sie können Astra Trident und alle erstellten CRDs und zugehörigen benutzerdefinierten Ressourcen vollständig entfernen.



Dieser Vorgang kann nicht rückgängig gemacht werden. Tun Sie dies nur, wenn Sie eine völlig frische Installation von Astra Trident wollen. Informationen zur Deinstallation von Astra Trident ohne Entfernen von CRDs finden Sie unter ["Deinstallieren Sie Astra Trident"](#).

Betreiber von Trident

So deinstallieren Sie Astra Trident und entfernen Sie CRDs vollständig mit dem Trident Operator:

```
kubectl patch torc <trident-orchestrator-name> --type=merge -p
'{"spec":{"wipeout":["crds"],"uninstall":true}}'
```

Helm

So deinstallieren Sie Astra Trident und entfernen Sie CRDs vollständig mit Helm:

```
kubectl patch torc trident --type=merge -p
'{"spec":{"wipeout":["crds"],"uninstall":true}}'
```

`tridentctl`

So entfernen Sie CRDs nach der Deinstallation von Astra Trident vollständig mit `tridentctl`

```
tridentctl obliviate crd
```

Unterstützung

NetApp bietet unterschiedliche Unterstützung für Astra Trident. Umfangreiche kostenlose Self-Support-Optionen stehen rund um die Uhr zur Verfügung, wie z. B. Knowledge Base-Artikel (KB) und ein Einseilkanal.

Astra Trident Support-Lebenszyklus

Astra Trident bietet auf Basis Ihrer Version drei Support-Level. Siehe ["Unterstützung der NetApp Softwareversion für Definitionen"](#).

Volle Unterstützung

Astra Trident bietet ab Veröffentlichungsdatum vollen Support für zwölf Monate.

Eingeschränkter Support

Astra Trident bietet eingeschränkten Support für die Monate 13 bis 24 ab Veröffentlichungsdatum.

Self-Support

Die Dokumentation zu Astra Trident ist für die Monate 25 bis 36 ab Veröffentlichungsdatum verfügbar.

| Version | Volle Unterstützung | Eingeschränkter Support | Self-Support |
|---------|---------------------|-------------------------|--------------|
| "23.07" | Juli 2024 | Juli 2025 | Juli 2026 |
| "23.04" | April 2024 | April 2025 | April 2026 |
| "23.01" | Januar 2024 | Januar 2025 | Januar 2026 |
| "22.10" | Oktober 2023 | Oktober 2024 | Oktober 2025 |
| "22.07" | — | Juli 2024 | Juli 2025 |
| "22.04" | — | April 2024 | April 2025 |
| "22.01" | — | Januar 2024 | Januar 2025 |
| "21.10" | — | Oktober 2023 | Oktober 2024 |
| "21.07" | — | — | Juli 2024 |
| "21.04" | — | — | April 2024 |
| "21.01" | — | — | Januar 2024 |
| "20.10" | — | — | Oktober 2023 |



Versionen vor 21.07, Umleitung zum Standort für ältere Dokumentation.

Self-Support

Eine umfassende Liste von Artikeln zur Fehlerbehebung finden Sie im ["NetApp Knowledge Base \(Anmeldung erforderlich\)"](#). Hier finden Sie auch Informationen zur Behebung von Problemen im Zusammenhang mit Astra ["Hier"](#).

Community-Support

Es gibt eine lebendige, öffentliche Community von Container-Benutzern (einschließlich Astra Trident Entwickler) auf unserem Astra ["Kanal abstecken"](#). Hier können Sie allgemeine Fragen zum Projekt stellen und verwandte Themen mit Gleichgesinnten diskutieren.

Technischer Support von NetApp

Astra Trident unterstützt Sie bei der Erstellung eines Support-Bundles mit `tridentctl logs -a -n trident` Senden an NetApp Support [<Getting Help>](#).

Finden Sie weitere Informationen

- ["Astra-Blogs"](#)
- ["Astra Trident Blogs"](#)
- ["Kubernetes Hub"](#)
- ["NetApp.io"](#)

Referenz

Astra Trident-Ports

Erfahren Sie mehr über die Kommunikationsports von Astra Trident.

Astra Trident-Ports

Astra Trident kommuniziert über folgende Ports:

| Port | Zweck |
|-------|--|
| 8443 | Backchannel HTTPS |
| 8001 | Endpunkt der Prometheus Kennzahlen |
| 8000 | Trident REST-Server |
| 17546 | Anschluss für Liveness/Readiness-Sonde, der von Trident Demonset-Pods verwendet wird |



Der Anschluss der Liveness/Readiness-Sonde kann während der Installation mit dem geändert werden `--probe-port` Flagge. Es ist wichtig, sicherzustellen, dass dieser Port nicht von einem anderen Prozess auf den Worker-Knoten verwendet wird.

Astra Trident REST-API

Während "[Tridentctl-Befehle und -Optionen](#)" Die einfachste Möglichkeit, mit der Astra Trident REST-API zu interagieren, können Sie den REST-Endpunkt direkt verwenden, wenn Sie es bevorzugen.

Wann die REST-API verwendet werden soll

REST-API ist nützlich für erweiterte Installationen, in denen Astra Trident als eigenständige Binärdatei in Implementierungen ohne Kubernetes genutzt wird.

Für höhere Sicherheit bietet der Astra Trident an REST API Ist standardmäßig auf localhost beschränkt, wenn in einem Pod ausgeführt wird. Um dieses Verhalten zu ändern, müssen Sie Astra Trident's einstellen `-address` Argument in seiner Pod-Konfiguration.

REST-API wird verwendet

Die API funktioniert wie folgt:

GET

- `GET <trident-address>/trident/v1/<object-type>`: Listet alle Objekte dieses Typs auf.
- `GET <trident-address>/trident/v1/<object-type>/<object-name>`: Erhält die Details des genannten Objekts.

POST

POST <trident-address>/trident/v1/<object-type>: Erzeugt ein Objekt des angegebenen Typs.

- Eine JSON-Konfiguration für das zu erstellende Objekt erforderlich. Informationen zu den einzelnen Objekttypen finden Sie unter [Link:tridentctl.html](#)[tridentctl Befehle und Optionen].
- Falls das Objekt bereits vorhanden ist, variiert das Verhalten: Back-Ends aktualisiert das vorhandene Objekt, während alle anderen Objekttypen den Vorgang nicht ausführen.

DELETE

DELETE <trident-address>/trident/v1/<object-type>/<object-name>: Löscht die benannte Ressource.



Es existieren weiterhin Volumes, die mit Back-Ends oder Storage-Klassen verbunden sind. Diese müssen separat gelöscht werden. Weitere Informationen finden Sie unter [Link:tridentctl.html](#)[tridentctl Befehle und Optionen].

Für Beispiele, wie diese APIs aufgerufen werden, geben Sie das Debug (-d) Flagge. Weitere Informationen finden Sie unter [Link:tridentctl.html](#)[tridentctl Befehle und Optionen].

Befehlszeilenoptionen

Astra Trident stellt verschiedene Befehlszeilenoptionen für den Trident Orchestrator bereit. Sie können diese Optionen verwenden, um Ihre Bereitstellung zu ändern.

Protokollierung

- `-debug`: Aktiviert die Debugging-Ausgabe.
- `-loglevel <level>`: Legt die Protokollierungsebene fest (Debug, info, warn, error, fatal). Standardmäßig Info.

Kubernetes

- `-k8s_pod`: Verwenden Sie diese Option oder `-k8s_api_server` Um die Kubernetes-Unterstützung zu aktivieren. Durch diese Einstellung verwendet Trident die Zugangsdaten für das Kubernetes-Servicekonto eines Pods, um den API-Server zu kontaktieren. Dies funktioniert nur, wenn Trident als Pod in einem Kubernetes-Cluster mit aktivierten Service-Konten ausgeführt wird.
- `-k8s_api_server <insecure-address:insecure-port>`: Verwenden Sie diese Option oder `-k8s_pod` Um die Kubernetes-Unterstützung zu aktivieren. Bei Angabe von stellt Trident über die angegebene unsichere Adresse und den angegebenen Port eine Verbindung zum Kubernetes-API-Server her. Dadurch kann Trident außerhalb eines Pods implementiert werden; es unterstützt jedoch nur unsichere Verbindungen zum API-Server. Mit der können Sie Trident sicher in einem Pod implementieren `-k8s_pod` Option.
- `-k8s_config_path <file>`: Erforderlich; Sie müssen diesen Pfad zu einer KubeConfig-Datei angeben.

Docker

- `-volume_driver <name>`: Treibername, der bei der Registrierung des Docker Plugins verwendet wird. Standardmäßig auf `netapp`.
- `-driver_port <port-number>`: Hören Sie auf diesem Port statt auf einen UNIX-Domänensockel.
- `-config <file>`: Erforderlich; Sie müssen diesen Pfad zu einer Backend-Konfigurationsdatei angeben.

RUHE

- `-address <ip-or-host>`: Gibt die Adresse an, auf der der REST-Server von Trident zuhören soll. Standardmäßig `localhost`. Wenn auf dem `localhost` zuhören und in einem Kubernetes Pod ausgeführt werden, ist der ZUGRIFF auf DIE REST-Schnittstelle nicht direkt von außerhalb des Pods möglich. Nutzung `-address ""` Damit die REST-Schnittstelle über die POD-IP-Adresse zugänglich ist.



Die Trident REST-Schnittstelle kann nur für die Wiedergabe unter 127.0.0.1 (für IPv4) oder `[: 1]` (für IPv6) konfiguriert werden.

- `-port <port-number>`: Gibt den Port an, auf dem der REST-Server von Trident zuhören soll. Die Standardeinstellung ist 8000.
- `-rest`: Aktiviert die REST-Schnittstelle. Standardmäßig auf „true“ gesetzt.

NetApp Produkte sind in Kubernetes integriert

Das NetApp Portfolio an Storage-Produkten lässt sich in viele verschiedene Aspekte eines Kubernetes Clusters integrieren und bietet erweiterte Datenmanagementfunktionen zur Verbesserung von Funktionen, Funktionen, Performance und Verfügbarkeit der Kubernetes-Implementierung.

Astra

"Astra" Unternehmen können ihre datenintensiven Container-Workloads, die innerhalb von Kubernetes ausgeführt werden, leichter managen, schützen und on-Premises verschieben. Astra stellt persistenten Container-Storage mithilfe von Trident aus dem bewährten und umfangreichen NetApp-Storage-Portfolio für Public Clouds und On-Premises bereit. Außerdem bietet es umfangreiche erweiterte, applikationsspezifische Datenmanagementfunktionen, wie Snapshot, Backup und Wiederherstellung, Aktivitätsprotokolle und aktives Klonen für Datensicherung, Disaster/Daten-Recovery, Datenaudits und Migrationsanwendungsfälle für Kubernetes-Workloads.

ONTAP

ONTAP ist das Unified Storage-Betriebssystem für Multi-Protokoll-Unterstützung von NetApp und bietet für jede Applikation erweiterte Datenmanagement-Funktionen. ONTAP Systeme verfügen über rein Flash-basierte, hybride oder rein HDD-basierte Konfigurationen und bieten eine Vielzahl unterschiedlicher Implementierungsmodelle, darunter speziell entwickelte Hardware (FAS und AFF), White-Box (ONTAP Select) und rein Cloud-basierte Cloud Volumes ONTAP Systeme.



Trident unterstützt alle oben genannten ONTAP Implementierungsmodelle.

Cloud Volumes ONTAP

"[Cloud Volumes ONTAP](#)" Ist eine rein softwarebasierte Storage Appliance, die die ONTAP Datenmanagement-Software in der Cloud ausführt. Sie können Cloud Volumes ONTAP für Produktions-Workloads, Disaster Recovery, DevOps, Dateifreigaben und Datenbankmanagement verwenden. Sie erweitert den Enterprise-Storage auf die Cloud und bietet Storage-Effizienz, Hochverfügbarkeit, Datenreplizierung, Daten-Tiering und Applikationskonsistenz.

Amazon FSX für NetApp ONTAP

"[Amazon FSX für NetApp ONTAP](#)" Ist ein vollständig gemanagter AWS Service, mit dem Sie Dateisysteme mit dem NetApp ONTAP Storage-Betriebssystem starten und ausführen können. FSX for ONTAP ermöglicht Ihnen die Nutzung von NetApp-Funktionen, Performance und Administrationsfunktionen, mit denen Sie vertraut sind, und gleichzeitig die Einfachheit, Agilität, Sicherheit und Skalierbarkeit der Speicherung von Daten auf AWS zu nutzen. FSX für ONTAP unterstützt viele der ONTAP-Dateisystemfunktionen und Administrations-APIs.

Element Software

"[Element](#)" Storage-Administrator kann Workloads konsolidieren, indem die Performance garantiert und der Storage-Bedarf vereinfacht und optimiert wird. Kombiniert mit einer API zur Automatisierung aller Aspekte des Storage-Managements unterstützt Element Storage-Administratoren dabei, mit weniger Aufwand mehr zu erreichen.

NetApp HCI

"[NetApp HCI](#)" Vereinfacht das Management und die Skalierung des Datacenters durch Automatisierung von Routineaufgaben und ermöglicht es Infrastrukturadministratoren, sich auf wichtigere Funktionen zu konzentrieren.

NetApp HCI wird vollständig von Trident unterstützt. Trident kann Storage-Geräte für Container-Applikationen direkt auf der zugrunde liegenden NetApp HCI Storage-Plattform bereitstellen und managen.

Azure NetApp Dateien

"[Azure NetApp Dateien](#)" Ist ein Azure-Dateifreigabeservice der Enterprise-Klasse auf der Basis von NetApp. Sie können anspruchsvollste dateibasierte Workloads nativ in Azure ausführen. So erhalten Sie die Performance und das umfassende Datenmanagement, die Sie von NetApp gewohnt sind.

Cloud Volumes Service für Google Cloud

"[NetApp Cloud Volumes Service für Google Cloud](#)" Ist ein Cloud-nativer Fileservice, der NAS-Volumes über NFS und SMB mit All-Flash-Performance bereitstellt. Mit diesem Service können alle Workloads, auch ältere Applikationen, in der GCP-Cloud ausgeführt werden. Es bietet einen vollständig gemanagten Service, der konsistent hohe Performance, sofortiges Klonen, Datensicherung und sicheren Zugriff auf Google Compute Engine (GCE) Instanzen bietet.

Kubernetes und Trident Objekte

Kubernetes und Trident lassen sich über REST-APIs miteinander interagieren, indem Objekte gelesen und geschrieben werden. Es gibt verschiedene Ressourcenobjekte, die die Beziehung zwischen Kubernetes und Trident, Trident und Storage sowie Kubernetes und Storage vorschreiben. Einige dieser Objekte werden über Kubernetes verwaltet,

andere wiederum über Trident.

Wie interagieren die Objekte miteinander?

Am einfachsten ist es, die Objekte, deren Bedeutung und ihre Interaktion zu verstehen, wenn ein Kubernetes-Benutzer eine einzelne Storage-Anfrage bearbeitet:

1. Ein Benutzer erstellt ein `PersistentVolumeClaim` Anforderung eines neuen `PersistentVolume` Einer bestimmten Größe von einem Kubernetes aus `StorageClass` Das wurde zuvor vom Administrator konfiguriert.
2. Kubernetes `StorageClass` Identifiziert Trident als seine bereitstellung und enthält Parameter, die Trident zur Bereitstellung eines Volumes für die angeforderte Klasse angeben.
3. Trident sieht seinen eigenen Blick `StorageClass` Mit dem gleichen Namen, der die Übereinstimmung identifiziert `Backends` Und `StoragePools` Die sie für die Bereitstellung von Volumes für die Klasse einsetzen kann.
4. Trident stellt Storage auf einem passenden Back-End bereit und erstellt zwei Objekte: A `PersistentVolume` In Kubernetes informiert Kubernetes über das Finden, Mounten und behandeln des Volumes und ein `Volume` in Trident, das die Beziehung zwischen den beibehält `PersistentVolume` Und dem tatsächlichen Storage.
5. Kubernetes bindet das `PersistentVolumeClaim` Zum neuen `PersistentVolume`. Pods, die die enthalten `PersistentVolumeClaim` Mounten Sie dieses `PersistenzVolume` auf jedem Host, auf dem es ausgeführt wird.
6. Ein Benutzer erstellt ein `VolumeSnapshot` Eines vorhandenen PVC unter Verwendung eines `VolumeSnapshotClass` Das verweist auf Trident.
7. Trident identifiziert das dem PVC zugeordnete `Volume` und erstellt einen Snapshot des Volumes auf dem Back-End. Es erzeugt auch ein `VolumeSnapshotContent` Damit wird Kubernetes angewiesen, den Snapshot zu identifizieren.
8. Ein Benutzer kann ein erstellen `PersistentVolumeClaim` Wird verwendet `VolumeSnapshot` Als Quelle.
9. Trident identifiziert den erforderlichen Snapshot und führt die gleichen Schritte aus, die bei der Erstellung eines erforderlich sind `PersistentVolume` Und A `Volume`.



Für weitere Informationen über Kubernetes-Objekte empfehlen wir Ihnen, die zu lesen "[Persistente Volumes](#)" Der Kubernetes-Dokumentation.

Kubernetes `PersistentVolumeClaim` Objekte

Ein Kubernetes `PersistentVolumeClaim` Objekt ist eine Storage-Anfrage von einem Kubernetes Cluster-Benutzer.

Zusätzlich zur Standardspezifikation können Benutzer mit Trident die folgenden Volume-spezifischen Anmerkungen angeben, wenn sie die in der Back-End-Konfiguration festgelegten Standardeinstellungen überschreiben möchten:

| Anmerkung | Volume-Option | Unterstützte Treiber |
|---|---------------|--|
| <code>trident.netapp.io/fileSystem</code> | Dateisystem | <code>ontap-san</code> , <code>solidfire-san</code> , <code>ontap-san-Economy</code> |

| Anmerkung | Volume-Option | Unterstützte Treiber |
|-------------------------------------|-------------------|---|
| trident.netapp.io/cloneFromPVC | KlonSourceVolume | ontap-nas ontap-san, solidfire-san, Azure-netapp-Files, gcp-cim, ontap-san-Ökonomie |
| trident.netapp.io/splitOnClone | SPLITOnClone | ontap-nas, ontap-san |
| trident.netapp.io/protocol | Protokoll | Alle |
| trident.netapp.io/exportPolicy | Exportpolitik | ontap-nas ontap-nas-Economy, ontap-nas-flexgroup |
| trident.netapp.io/snapshotPolicy | SnapshotPolicy | ontap-nas ontap-nas-Economy, ontap-nas-flexgroup, ontap-san |
| trident.netapp.io/snapshotReserve | SnapshotReserve | ontap-nas ontap-nas-flexgroup, ontap-san, gcp-cvs |
| trident.netapp.io/snapshotDirectory | SnapshotDirectory | ontap-nas ontap-nas-Economy, ontap-nas-flexgroup |
| trident.netapp.io/unixPermissions | UnxPermissions | ontap-nas ontap-nas-Economy, ontap-nas-flexgroup |
| trident.netapp.io/blockSize | Blocksize | solidfire-san |

Wenn das erstellte PV über den verfügt `Delete` Rückgewinnungsrichtlinie: Trident löscht sowohl das PV als auch das Backvolume, wenn das PV freigegeben wird (d. h. wenn der Benutzer die PVC löscht). Sollte die Löschkaktion fehlschlagen, markiert Trident den PV als solche und wiederholt den Vorgang periodisch, bis er erfolgreich ist oder der PV manuell gelöscht wird. Wenn das PV den verwendet `Retain` Richtlinie: Trident ignoriert es und geht davon aus, dass der Administrator die Datei über Kubernetes und das Backend bereinigt, damit das Volume vor dem Entfernen gesichert oder inspiziert werden kann. Beachten Sie, dass das Löschen des PV nicht dazu führt, dass Trident das Backing-Volume löscht. Sie sollten es mit DER REST API entfernen (`tridentctl`).

Trident unterstützt die Erstellung von Volume Snapshots anhand der CSI-Spezifikation: Sie können einen Volume Snapshot erstellen und ihn als Datenquelle zum Klonen vorhandener PVCs verwenden. So können zeitpunktgenaue Kopien von PVS in Form von Snapshots Kubernetes zugänglich gemacht werden. Die Snapshots können dann verwendet werden, um neue PVS zu erstellen. Sie finden sie hier `On-Demand Volume Snapshots` Um zu sehen, wie das funktionieren würde.

Trident enthält außerdem die `cloneFromPVC` Und `splitOnClone` Anmerkungen zum Erstellen von Klonen. Mit diesen Anmerkungen können Sie eine PVC klonen, ohne die CSI-Implementierung verwenden zu müssen.

Hier ist ein Beispiel: Wenn ein Benutzer bereits ein PVC aufgerufen hat `mysql`, Der Benutzer kann ein neues PVC mit dem Namen erstellen `mysqlclone` Durch die Verwendung der Anmerkung, z. B.

`trident.netapp.io/cloneFromPVC: mysql`. Mit diesem Anmerkungsset klonst Trident das Volume, das dem `mysql` PVC entspricht, anstatt ein Volume von Grund auf neu bereitzustellen.

Berücksichtigen Sie folgende Punkte:

- Wir empfehlen das Klonen eines inaktiven Volumes.
- Ein PVC und sein Klon sollten sich im gleichen Kubernetes Namespace befinden und dieselbe Storage-Klasse haben.
- Mit dem `ontap-nas` Und `ontap-san` Treiber, kann es wünschenswert sein, die PVC-Anmerkung zu setzen `trident.netapp.io/splitOnClone` Zusammen mit `trident.netapp.io/cloneFromPVC`. Mit `trident.netapp.io/splitOnClone` Auf einstellen `true`, Trident teilt das geklonte Volume vom übergeordneten Volume auf und sorgt so für eine vollständige Entkopplung des geklonten Volume vom übergeordneten Volume – und zwar auf Kosten des Verlusts von Storage-Effizienz. Keine Einstellung `trident.netapp.io/splitOnClone` Oder auf einstellen `false` Dies senkt den Platzbedarf im Back-End. Dies verursacht Abhängigkeiten zwischen dem übergeordneten und den Klon-Volumes, sodass das übergeordnete Volume nur gelöscht werden kann, wenn der Klon zuvor gelöscht wird. Ein Szenario, in dem das Aufteilen des Klons sinnvoll ist, ist das Klonen eines leeren Datenbank-Volumes, in dem erwartet wird, dass das Volume und der zugehörige Klon eine große Divergenz sind. Es profitieren nicht von der Storage-Effizienz des ONTAP.

Der `sample-input` Das Verzeichnis enthält Beispiele für PVC-Definitionen zur Verwendung mit Trident. Siehe Eine vollständige Beschreibung der Parameter und Einstellungen zu Trident Volumes.

Kubernetes PersistentVolume Objekte

Ein Kubernetes `PersistentVolume` Objekt stellt eine Storage-Komponente dar, die dem Kubernetes-Cluster zur Verfügung gestellt wird. Es weist einen Lebenszyklus auf, der unabhängig vom POD ist, der ihn nutzt.



Trident erstellt `PersistentVolume` Objekte werden beim Kubernetes Cluster automatisch auf Basis der Volumes registriert, die bereitgestellt werden. Sie sollten diese nicht selbst verwalten.

Wenn Sie eine PVC erstellen, die sich auf eine Trident-basierte bezieht `StorageClass`, Trident stellt ein neues Volume anhand der entsprechenden Storage-Klasse bereit und registriert ein neues PV für dieses Volume. Bei der Konfiguration des bereitgestellten Volume und des entsprechenden PV befolgt Trident folgende Regeln:

- Trident generiert einen PV-Namen für Kubernetes mit einem internen Namen, der zur Bereitstellung des Storage verwendet wird. In beiden Fällen wird sichergestellt, dass die Namen in ihrem Geltungsbereich eindeutig sind.
- Die Größe des Volumens entspricht der gewünschten Größe in der PVC so genau wie möglich, obwohl es möglicherweise auf die nächste zuteilbare Menge aufgerundet werden, je nach Plattform.

Kubernetes StorageClass Objekte

Kubernetes `StorageClass` Objekte werden in mit Namen angegeben `PersistentVolumeClaims` So stellen Sie Speicher mit einer Reihe von Eigenschaften bereit. Die Storage-Klasse selbst gibt die zu verwendenden bereitstellungsunternehmen an und definiert die Eigenschaftengruppe in Bezug auf die provisionierung von.

Es handelt sich um eines von zwei grundlegenden Objekten, die vom Administrator erstellt und verwaltet werden müssen. Das andere ist das Trident Back-End-Objekt.

Ein Kubernetes `StorageClass` Objekt, das Trident verwendet, sieht so aus:

```

apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: <Name>
provisioner: csi.trident.netapp.io
mountOptions: <Mount Options>
parameters:
  <Trident Parameters>
allowVolumeExpansion: true
volumeBindingMode: Immediate

```

Diese Parameter sind Trident-spezifisch und Trident erläutert die Bereitstellung von Volumes für die Klasse.

Parameter der Storage-Klasse sind:

| Attribut | Typ | Erforderlich | Beschreibung |
|---------------------------------|-------------------------------------|--------------|---|
| Merkmale | Zuordnen einer Zeichenfolge[string] | Nein | Weitere Informationen finden Sie im Abschnitt Attribute unten |
| Storage Pools | Zuordnen[String]StringList | Nein | Zuordnung von Backend-Namen zu Listen Storage-Pools in NetApp zu nutzen |
| Zusätzlich StoragePools | Zuordnen[String]StringList | Nein | Zuordnung der Backend-Namen Listen von Speicherpools in |
| Unter Ausnahme von StoragePools | Zuordnen[String]StringList | Nein | Zuordnung der Backend-Namen zu Listen der Speicherpools in |

Storage-Attribute und ihre möglichen Werte können in Auswahlebene und Kubernetes-Attribute des Storage-Pools klassifiziert werden.

Auswahlebene für Storage-Pools

Diese Parameter bestimmen, welche in Trident gemanagten Storage Pools zur Bereitstellung von Volumes eines bestimmten Typs verwendet werden sollten.

| Attribut | Typ | Werte | Angebot | Anfrage | Unterstützt von |
|---------------------|--------------|------------------|--|-------------------|---|
| Medien ¹ | Zeichenfolge | hdd, Hybrid, ssd | Pool enthält Medien dieser Art. Beides bedeutet Hybrid | Medientyp angeben | ontap-nas, ontap-nas-Economy, ontap-nas-Flexgroup, ontap-san, solidfire-san |

| Attribut | Typ | Werte | Angebot | Anfrage | Unterstützt von |
|--------------------|--------------|--|--|--|---|
| Bereitstellungstyp | Zeichenfolge | Dünn, dick | Pool unterstützt diese Bereitstellungs-methode | Bereitstellungsmethode angegeben | Thick: All ONTAP; Thin: Alle ONTAP und solidfire-san |
| BackendType | Zeichenfolge | ontap-nas, ontap-nas-Economy, ontap-nas-Flexgroup, ontap-san, solidfire-san, gcp-cvs, Azure-netapp-Files, ontap-san-Wirtschaftlichkeit | Pool gehört zu dieser Art von Backend | Back-End angegeben | Alle Treiber |
| Snapshots | bool | Richtig, falsch | Pool unterstützt Volumes mit Snapshots | Volume mit aktivierten Snapshots | ontap-nas, ontap-san, solidfire-san, gcp-cvs |
| Klone | bool | Richtig, falsch | Pool unterstützt das Klonen von Volumes | Volume mit aktivierten Klone | ontap-nas, ontap-san, solidfire-san, gcp-cvs |
| Verschlüsselung | bool | Richtig, falsch | Pool unterstützt verschlüsselte Volumes | Volume mit aktivierter Verschlüsselung | ontap-nas, ontap-nas-Economy, ontap-nas-Flexgroups, ontap-san |
| IOPS | Int | Positive Ganzzahl | Pool kann IOPS in diesem Bereich garantieren | Volume hat diese IOPS garantiert | solidfire-san |

¹: Nicht unterstützt von ONTAP Select-Systemen

In den meisten Fällen beeinflussen die angeforderten Werte direkt die Bereitstellung. Wenn Sie beispielsweise Thick Provisioning anfordern, entsteht ein Volume mit Thick Provisioning. Ein Element Storage-Pool nutzt jedoch den angebotenen IOPS-Minimum und das Maximum, um QoS-Werte anstelle des angeforderten Werts festzulegen. In diesem Fall wird der angeforderte Wert nur verwendet, um den Speicherpool auszuwählen.

Im Idealfall können Sie verwenden `attributes` Um die Eigenschaften des Storage zu modellieren, können Sie die Anforderungen einer bestimmten Klasse erfüllen. Trident erkennt und wählt automatisch Storage Pools aus, die mit `all` der übereinstimmen `attributes` Die Sie angeben.

Wenn Sie feststellen, dass Sie nicht in der Lage sind, zu verwenden `attributes` Um automatisch die richtigen Pools für eine Klasse auszuwählen, können Sie die verwenden `storagePools` Und `additionalStoragePools` Parameter zur weiteren Verfeinerung der Pools oder sogar zur Auswahl einer bestimmten Gruppe von Pools.

Sie können das verwenden `storagePools` Parameter zur weiteren Einschränkung des Pools, die mit den angegebenen übereinstimmen `attributes`. Mit anderen Worten: Trident verwendet die Schnittstelle von Pools, die vom identifiziert werden `attributes` Und `storagePools` Parameter für die Bereitstellung. Sie können entweder allein oder beides zusammen verwenden.

Sie können das verwenden `additionalStoragePools` Parameter zur Erweiterung des Pools, die Trident für die Bereitstellung verwendet, unabhängig von den vom ausgewählten Pools `attributes` Und `storagePools` Parameter.

Sie können das verwenden `excludeStoragePools` Parameter zum Filtern des Pools, den Trident für die Bereitstellung verwendet. Mit diesem Parameter werden alle Pools entfernt, die übereinstimmen.

Im `storagePools` Und `additionalStoragePools` Parameter, jeder Eintrag nimmt das Formular `<backend>:<storagePoolList>`, Wo `<storagePoolList>` Ist eine kommasetrennte Liste von Speicherpools für das angegebene Backend. Beispiel: Ein Wert für `additionalStoragePools` Könnte aussehen `ontapnas_192.168.1.100:aggr1,aggr2;solidfire_192.168.1.101:bronze`. Diese Listen akzeptieren Regex-Werte sowohl für das Backend als auch für Listenwerte. Verwenden Sie können `tridentctl get backend` Um die Liste der Back-Ends und deren Pools zu erhalten.

Attribute für Kubernetes

Diese Attribute haben keine Auswirkung auf die Auswahl von Storage-Pools/Back-Ends, die von Trident während der dynamischen Provisionierung durchgeführt werden. Stattdessen liefern diese Attribute einfach Parameter, die von Kubernetes Persistent Volumes unterstützt werden. Worker-Knoten sind für die Erstellung von Dateisystem-Operationen verantwortlich und benötigen möglicherweise Dateisystem-Dienstprogramme, wie z. B. `xfspgs`.

| Attribut | Typ | Werte | Beschreibung | Wichtige Faktoren | Kubernetes Version |
|-------------------|--------------|----------------------|---|---|--------------------|
| Fstype | Zeichenfolge | Ext4, ext3, xfs usw. | Der Dateisystemtyp für Block Volumes | solidfire-san, ontap-nas, ontap-nas-Economy, ontap-nas-Flexgroup, ontap-san, ontap-san-Ökonomie | Alle |
| VolumeErweiterung | boolesch | Richtig, falsch | Aktivieren oder deaktivieren Sie die Unterstützung für das Vergrößern der PVC-Größe | ontap-nas, ontap-nas-Ökonomie, ontap-nas-Flexgroup, ontap-san, ontap-san-Ökonomie, solidfire-san, gcp-cvs, Azure-netapp-Files | 1.11 und höher |

| | | | | | |
|--------------------|--------------|------------------------------|---|------|-------------|
| VolumeBindingmodus | Zeichenfolge | Sofort, WaitForFirstConsumer | Legen Sie fest, wann Volume Binding und dynamische Bereitstellung stattfindet | Alle | 1.19 - 1.26 |
|--------------------|--------------|------------------------------|---|------|-------------|

- Der `fsType` Parameter wird verwendet, um den gewünschten Filesystem-Typ für SAN-LUNs zu steuern. Darüber hinaus verwendet Kubernetes auch Präsenz von `fsType` In einer Speicherklasse, die darauf hinweist, dass ein Dateisystem vorhanden ist. Das Volume-Eigentum kann über den gesteuert werden `fsGroup` Sicherheitskontext eines Pods nur wenn `fsType` Ist festgelegt. Siehe ["Kubernetes: Einen Sicherheitskontext für einen Pod oder Container konfigurieren"](#) Für eine Übersicht über die Einstellung des Volume-Besitzes mit dem `fsGroup` Kontext. Kubernetes wendet das an `fsGroup` Wert nur, wenn:

- `fsType` Wird in der Storage-Klasse festgelegt.
- Der PVC-Zugriffsmodus ist `RWO`.



Für NFS-Speichertreiber ist bereits ein Dateisystem als Teil des NFS-Exports vorhanden. Zur Verwendung `fsGroup` Die Storage-Klasse muss noch ein angeben `fsType`. Sie können es auf einstellen `nfs` Oder ein nicht-Null-Wert.

- Siehe ["Erweitern Sie Volumes"](#) Für weitere Informationen zur Volume-Erweiterung.
- Das Trident Installationspaket bietet verschiedene Beispiele für Storage-Klassen, die mit Trident in verwendet werden können `sample-input/storage-class-*.yaml`. Durch das Löschen einer Kubernetes-Storage-Klasse wird auch die entsprechende Trident-Storage-Klasse gelöscht.

Kubernetes VolumeSnapshotClass Objekte

Kubernetes `VolumeSnapshotClass` Objekte sind analog `StorageClasses`. Sie helfen, mehrere Speicherklassen zu definieren und werden von Volume-Snapshots referenziert, um den Snapshot der erforderlichen Snapshot-Klasse zuzuordnen. Jeder Volume Snapshot ist einer einzelnen Volume-Snapshot-Klasse zugeordnet.

A `VolumeSnapshotClass` Sollte von einem Administrator definiert werden, um Snapshots zu erstellen. Eine Volume-Snapshot-Klasse wird mit folgender Definition erstellt:

```
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshotClass
metadata:
  name: csi-snapclass
driver: csi.trident.netapp.io
deletionPolicy: Delete
```

Der `driver` Gibt an Kubernetes, dass Volume-Snapshots von anfordert `csi-snapclass` Die Klasse werden von Trident übernommen. Der `deletionPolicy` Gibt die Aktion an, die ausgeführt werden soll, wenn ein Snapshot gelöscht werden muss. Wenn `deletionPolicy` Ist auf festgelegt `Delete`, Die Volume-Snapshot-Objekte sowie der zugrunde liegende Snapshot auf dem Storage-Cluster werden entfernt, wenn ein Snapshot

gelöscht wird. Alternativ können Sie ihn auf `retain` einstellen. `retain` bedeutet, dass das `VolumeSnapshotContent` und der physische Snapshot beibehalten werden.

Kubernetes `VolumeSnapshot` Objekte

Ein Kubernetes `VolumeSnapshot` Objekt ist eine Anforderung zur Erstellung eines Snapshots eines Volumes. So wie eine PVC eine von einem Benutzer erstellte Anfrage für ein Volume darstellt, besteht bei einem `VolumeSnapshot` die Anforderung eines Benutzers, einen Snapshot eines vorhandenen PVC zu erstellen.

Sobald eine `VolumeSnapshot`-Anfrage eingeht, managt Trident automatisch die Erstellung des Snapshots für das Volume auf dem Backend und legt den Snapshot offen, indem er einen eindeutigen `VolumeSnapshotContent` Objekt erstellt. Sie können Snapshots aus vorhandenen VES erstellen und die Snapshots als Datenquelle beim Erstellen neuer VES verwenden.



Der Lebenszyklus eines `VolumeSnapshot` ist unabhängig von der Quelle PVC: Ein Snapshot bleibt auch nach dem Löschen der Quelle PVC erhalten. Beim Löschen eines PVC mit zugehörigen Snapshots markiert Trident das Backing-Volume für dieses PVC in einem **Deleting**-Zustand, entfernt es aber nicht vollständig. Das Volume wird entfernt, wenn alle zugehörigen Snapshots gelöscht werden.

Kubernetes `VolumeSnapshotContent` Objekte

Ein Kubernetes `VolumeSnapshotContent` Objekt stellt einen Snapshot dar, der von einem bereits bereitgestellten Volume entnommen wurde. Es ist analog zu einem `PersistentVolume` und bedeutet einen bereitgestellten Snapshot auf dem Storage-Cluster. Ähnlich `PersistentVolumeClaim` und `PersistentVolume` Objekte, wenn ein Snapshot erstellt wird, das `VolumeSnapshotContent` Objekt verwaltet eine 1:1-Zuordnung zum `VolumeSnapshot` Objekt, das die Snapshot-Erstellung angefordert hatte.

Das `VolumeSnapshotContent` Objekt enthält Details, die den Snapshot eindeutig identifizieren, z. B. den `snapshotHandle`. Das `snapshotHandle` ist eine einzigartige Kombination aus dem Namen des PV und dem Namen des `VolumeSnapshotContent` Objekt:

Wenn eine Snapshot-Anfrage eingeht, erstellt Trident den Snapshot auf dem Back-End. Nach der Erstellung des Snapshots konfiguriert Trident ein `VolumeSnapshotContent` Objekt-Storage erstellt und damit den Snapshot der Kubernetes API zur Verfügung gestellt.



In der Regel müssen Sie das nicht verwalten `VolumeSnapshotContent` Objekt: Eine Ausnahme ist, wenn Sie möchten **"Importieren Sie einen Volume-Snapshot"** Erstellen außerhalb von Astra Trident.

Kubernetes `CustomResourceDefinition` Objekte

Kubernetes Custom Ressourcen sind Endpunkte in der Kubernetes API, die vom Administrator definiert werden und zum Gruppieren ähnlicher Objekte verwendet werden. Kubernetes unterstützt das Erstellen individueller Ressourcen zum Speichern einer Sammlung von Objekten. Sie erhalten diese Ressourcen-Definitionen, indem Sie ausführen `kubectl get crds`.

CRDs (Custom Resource Definitions) und die zugehörigen Objektmetadaten werden durch Kubernetes im Metadaten Speicher gespeichert. Dadurch ist kein separater Speicher für Trident erforderlich.

Astra Trident verwendet `CustomResourceDefinition` Objekte zur Wahrung der Identität von Trident

Objekten, wie Trident Back-Ends, Trident Storage-Klassen und Trident Volumes. Diese Objekte werden von Trident gemanagt. Darüber hinaus werden im CSI-Volume-Snapshot-Framework einige CRS-IDs verwendet, die zum Definieren von Volume-Snapshots erforderlich sind.

CRDs stellen ein Kubernetes-Konstrukt dar. Objekte der oben definierten Ressourcen werden von Trident erstellt. Wenn ein Backend mit erstellt wird, ist das ein einfaches Beispiel `tridentctl`, Eine entsprechende `tridentbackends` Das CRD-Objekt wird für den Verbrauch durch Kubernetes erstellt.

Beachten Sie die folgenden CRDs von Trident:

- Wenn Trident installiert ist, werden eine Reihe von CRDs erstellt und können wie alle anderen Ressourcentypen verwendet werden.
- Bei der Deinstallation von Trident mit dem `tridentctl uninstall` Befehl, Trident Pods werden gelöscht, die erstellten CRDs werden jedoch nicht bereinigt. Siehe "[Deinstallieren Sie Trident](#)" Um zu erfahren, wie Trident vollständig entfernt und von Grund auf neu konfiguriert werden kann

Astra Trident StorageClass Objekte

Trident erstellt passende Storage-Klassen für Kubernetes StorageClass Objekte, die angeben `csi.trident.netapp.io/netapp.io/trident` In ihrem Feld für die bereitstellung. Der Name der Storage-Klasse stimmt mit der der von Kubernetes überein StorageClass Objekt, das es repräsentiert.



Mit Kubernetes werden diese Objekte automatisch bei einem Kubernetes erstellt StorageClass Und Trident ist für die bereitstellung registriert.

Storage-Klassen umfassen eine Reihe von Anforderungen für Volumes. Trident stimmt diese Anforderungen mit den in jedem Storage-Pool vorhandenen Attributen überein. Ist dieser Storage-Pool ein gültiges Ziel für die Bereitstellung von Volumes anhand dieser Storage-Klasse.

Sie können Storage-Klassen-Konfigurationen erstellen, um Storage-Klassen direkt über DIE REST API zu definieren. Bei Kubernetes-Implementierungen werden sie jedoch bei der Registrierung von neuem Kubernetes erstellt StorageClass Objekte:

Back-End-Objekte für Astra Trident

Back-Ends stellen die Storage-Anbieter dar, über die Trident Volumes bereitstellt. Eine einzelne Trident Instanz kann eine beliebige Anzahl von Back-Ends managen.



Dies ist einer der beiden Objekttypen, die Sie selbst erstellen und verwalten. Die andere ist Kubernetes StorageClass Objekt:

Weitere Informationen zum Erstellen dieser Objekte finden Sie unter "[Back-Ends werden konfiguriert](#)".

Astra Trident StoragePool Objekte

Storage-Pools stellen die verschiedenen Standorte dar, die für die Provisionierung an jedem Back-End verfügbar sind. Für ONTAP entsprechen diese Aggregaten in SVMs. Bei NetApp HCI/SolidFire entsprechen diese den vom Administrator festgelegten QoS-Bands. Für Cloud Volumes Service entsprechen diese Regionen Cloud-Provider. Jeder Storage-Pool verfügt über eine Reihe individueller Storage-Attribute, die seine Performance-Merkmale und Datensicherungsmerkmale definieren.

Im Gegensatz zu den anderen Objekten hier werden Storage-Pool-Kandidaten immer automatisch erkannt und

gemanagt.

Astra Trident Volume Objekte

Volumes sind die grundlegende Bereitstellungseinheit, die Back-End-Endpunkte umfasst, wie NFS-Freigaben und iSCSI-LUNs. In Kubernetes entsprechen diese direkt `PersistentVolumes`. Wenn Sie ein Volume erstellen, stellen Sie sicher, dass es über eine Storage-Klasse verfügt, die bestimmt, wo das Volume zusammen mit einer Größe bereitgestellt werden kann.



- In Kubernetes werden diese Objekte automatisch gemanagt. Sie können sich anzeigen lassen, welche Bereitstellung von Trident bereitgestellt wurde.
- Wenn Sie ein PV mit den zugehörigen Snapshots löschen, wird das entsprechende Trident-Volume auf den Status **Löschen** aktualisiert. Damit das Trident Volume gelöscht werden kann, sollten Sie die Snapshots des Volume entfernen.

Eine Volume-Konfiguration definiert die Eigenschaften, über die ein bereitgestelltes Volume verfügen sollte.

| Attribut | Typ | Erforderlich | Beschreibung |
|------------------|--------------|--------------|--|
| Version | Zeichenfolge | Nein | Version der Trident API („1“) |
| Name | Zeichenfolge | ja | Name des zu erstellenden Volumes |
| Storage Class | Zeichenfolge | ja | Storage-Klasse, die bei der Bereitstellung des Volumes verwendet werden muss |
| Größe | Zeichenfolge | ja | Größe des Volumes, das in Byte bereitgestellt werden soll |
| Protokoll | Zeichenfolge | Nein | Zu verwendenden Protokolltyp; „Datei“ oder „Block“ |
| InternalName | Zeichenfolge | Nein | Name des Objekts auf dem Storage-System, das von Trident generiert wird |
| KlonSourceVolume | Zeichenfolge | Nein | ONTAP (nas, san) & SolidFire-*: Name des Volumes aus dem geklont werden soll |
| SPlitOnClone | Zeichenfolge | Nein | ONTAP (nas, san): Den Klon von seinem übergeordneten Objekt trennen |
| SnapshotPolicy | Zeichenfolge | Nein | ONTAP-*: Die Snapshot-Richtlinie zu verwenden |

| Attribut | Typ | Erforderlich | Beschreibung |
|-------------------|--------------|--------------|---|
| SnapshotReserve | Zeichenfolge | Nein | ONTAP-*: Prozentsatz des für Schnappschüsse reservierten Volumens |
| Exportpolitik | Zeichenfolge | Nein | ontap-nas*: Richtlinie für den Export zu verwenden |
| SnapshotDirectory | bool | Nein | ontap-nas*: Ob das Snapshot-Verzeichnis sichtbar ist |
| UnxPermissions | Zeichenfolge | Nein | ontap-nas*: Anfängliche UNIX-Berechtigungen |
| Blocksize | Zeichenfolge | Nein | SolidFire-*: Block-/Sektorgröße |
| Dateisystem | Zeichenfolge | Nein | Typ des Filesystems |

Trident generiert `internalName` Beim Erstellen des Volumes. Dies besteht aus zwei Schritten. Zuerst wird das Speicherpräfix (entweder der Standard) voreingestellt `trident` Oder das Präfix in der Backend-Konfiguration) zum Volume-Namen, was zu einem Namen des Formulars führt `<prefix>-<volume-name>`. Anschließend wird der Name desinifiziert und die im Backend nicht zulässigen Zeichen ersetzt. Bei ONTAP Back-Ends werden Bindestriche mit Unterstriche ersetzt (d. h., der interne Name wird aus `<prefix>_<volume-name>`). Bei Element-Back-Ends werden Unterstriche durch Bindestriche ersetzt.

Sie können Volume-Konfigurationen verwenden, um Volumes direkt über DIE REST-API bereitzustellen. In Kubernetes-Implementierungen gehen die meisten Benutzer jedoch davon aus, den Standard Kubernetes zu verwenden `PersistentVolumeClaim` Methode. Trident erstellt dieses Volume-Objekt automatisch im Rahmen der Bereitstellung Prozess.

Astra Trident Snapshot Objekte

Snapshots sind eine zeitpunktgenaue Kopie von Volumes, die zur Bereitstellung neuer Volumes oder für Restores verwendet werden kann. In Kubernetes entsprechen diese direkt `VolumeSnapshotContent` Objekte: Jeder Snapshot ist einem Volume zugeordnet, das die Quelle der Daten für den Snapshot ist.

Beide `Snapshot` Objekt enthält die unten aufgeführten Eigenschaften:

| Attribut | Typ | Erforderlich | Beschreibung |
|--------------|--------------|--------------|--|
| Version | Zeichenfolge | Ja. | Version der Trident API („1“) |
| Name | Zeichenfolge | Ja. | Name des Trident Snapshot-Objekts |
| InternalName | Zeichenfolge | Ja. | Name des Trident Snapshot-Objekts auf dem Storage-System |
| VolumeName | Zeichenfolge | Ja. | Name des Persistent Volume, für das der Snapshot erstellt wird |

| Attribut | Typ | Erforderlich | Beschreibung |
|--------------------|--------------|--------------|--|
| VolumeInternalName | Zeichenfolge | Ja. | Name des zugehörigen Trident-Volume-Objekts auf dem Storage-System |



In Kubernetes werden diese Objekte automatisch gemanagt. Sie können sich anzeigen lassen, welche Bereitstellung von Trident bereitgestellt wurde.

Wenn ein Kubernetes `VolumeSnapshot` Objektanforderung wird erstellt, Trident erstellt ein Snapshot-Objekt auf dem zugrunde gelegten Storage-System. Der `internalName` Dieses Snapshot-Objekt wird durch Kombination des Präfixes generiert `snapshot-` Mit dem UID Des `VolumeSnapshot` Objekt (z. B. `snapshot-e8d8a0ca-9826-11e9-9807-525400f3f660`). `volumeName` Und `volumeInternalName` Werden ausgefüllt, indem die Details des Backing abgerufen werden
Datenmenge:

Astra Trident `ResourceQuota` Objekt

Das Trident-Eintreten verbraucht einen `system-node-critical` Priority Class – die in Kubernetes verfügbare Class mit höchster Priorität, damit Astra Trident Volumes beim ordnungsgemäßen Shutdown von Nodes identifizieren und bereinigen kann und Trident Demonset-Pods zulassen kann, dass Workloads mit niedriger Priorität in Clustern mit hohen Ressourcenbelastungen vorbeugen.

Astra Trident setzt hierfür ein `ResourceQuota` Möchten Sie sicherstellen, dass eine „System-Node-kritische“ Prioritätsklasse auf dem Trident-Demonset erfüllt ist. Vor der Implementierung und der Erstellung von Dämonen sucht Astra Trident die `ResourceQuota` Objekt und, falls nicht erkannt, wendet es an.

Wenn Sie mehr Kontrolle über das standardmäßige Ressourcenkontingent und die Prioritätsklasse benötigen, können Sie ein generieren `custom.yaml` Oder konfigurieren Sie die `ResourceQuota` Objekt mit Helm-Diagramm.

Im Folgenden finden Sie ein Beispiel für ein `ResourceQuota` Objekt mit Priorität des Trident-Dämonenset.

```
apiVersion: <version>
kind: ResourceQuota
metadata:
  name: trident-csi
  labels:
    app: node.csi.trident.netapp.io
spec:
  scopeSelector:
    matchExpressions:
      - operator : In
        scopeName: PriorityClass
        values: ["system-node-critical"]
```

Weitere Informationen zu Ressourcenkontingenten finden Sie unter ["Kubernetes: Ressourcenkontingente"](#).

Bereinigung ResourceQuota Wenn die Installation fehlschlägt

In seltenen Fällen, in denen die Installation nach dem fehlschlägt ResourceQuota Das Objekt wird erstellt, versuchen Sie es zuerst "[Deinstallation](#)" Und installieren Sie dann neu.

Wenn das nicht funktioniert, entfernen Sie manuell das ResourceQuota Objekt:

Entfernen ResourceQuota

Wenn Sie die eigene Ressourcenzuweisung steuern möchten, können Sie den Astra Trident entfernen ResourceQuota Objekt mit dem Befehl:

```
kubectl delete quota trident-csi -n trident
```

Tridentctl-Befehle und -Optionen

Der "[Trident Installationspaket](#)" Enthält ein Befehlszeilendienstprogramm, `tridentctl`, Das ist ein einfacher Zugriff auf Astra Trident. Kubernetes-Benutzer mit ausreichenden Berechtigungen können es verwenden, um Astra Trident zu installieren und direkt damit zu interagieren, um den Namespace zu managen, der den Astra Trident Pod enthält.

Verfügbare Befehle und Optionen

Führen Sie zur Verwendung Informationen aus `tridentctl --help`.

Die verfügbaren Befehle und globalen Optionen sind:

```
Usage:
  tridentctl [command]
```

Verfügbare Befehle:

- `create`: Fügen Sie eine Ressource zu Astra Trident hinzu.
- `delete`: Entfernen Sie eine oder mehrere Ressourcen von Astra Trident.
- `get`: Holen Sie sich eines oder mehrere Ressourcen von Astra Trident.
- `help`: Hilfe zu jedem Befehl.
- `images`: Drucken Sie eine Tabelle der Container-Bilder, die Astra Trident braucht.
- `import`: Import einer vorhandenen Ressource zu Astra Trident.
- `install`: Installation Astra Trident:
- `logs`: Drucken Sie die Protokolle von Astra Trident.
- `send`: Senden Sie eine Ressource von Astra Trident.
- `uninstall`: Astra Trident Deinstallieren.

- `update`: Ändern Sie eine Ressource in Astra Trident.
- `upgrade`: Upgrade einer Ressource in Astra Trident.
- `version`: Drucken Sie die Version von Astra Trident.

Markierungen:

- ``-d, --debug`: Debug-Ausgabe.
- ``-h, --help`: Hilfe für `tridentctl`.
- ``-n, --namespace string`: Namespace für Astra Trident Implementierung.
- ``-o, --output string`: Ausgabeformat. Einer von `json` `yaml`-Namen natürlich Ärmellos (Standard).
- ``-s, --server string`: Adresse/Port des Astra Trident REST Interface.



Die Trident REST-Schnittstelle kann nur für die Wiedergabe unter 127.0.0.1 (für IPv4) oder `[: 1]` (für IPv6) konfiguriert werden.



Die Trident REST-Schnittstelle kann nur für die Wiedergabe unter 127.0.0.1 (für IPv4) oder `[: 1]` (für IPv6) konfiguriert werden.

`create`

Sie können den ausführen `create` Befehl zum Hinzufügen einer Ressource zum Astra Trident.

```
Usage:
  tridentctl create [option]
```

Verfügbare Option:

`backend`: Fügen Sie ein Backend zu Astra Trident hinzu.

`delete`

Sie können die ausführen `delete` Befehl, um eine oder mehrere Ressourcen aus Astra Trident zu entfernen.

```
Usage:
  tridentctl delete [option]
```

Verfügbare Optionen:

- `backend`: Löschen Sie ein oder mehrere Storage-Back-Ends von Astra Trident.
- `snapshot`: Löschen Sie einen oder mehrere Volumen-Snapshots aus Astra Trident.
- `storageclass`: Löschen einer oder mehrerer Speicherklassen von Astra Trident.
- `volume`: Löschen Sie ein oder mehrere Storage Volumes von Astra Trident.

get

Sie können die ausführen `get` Befehl: Sie erhalten eine oder mehrere Ressourcen von Astra Trident.

```
Usage:
  tridentctl get [option]
```

Verfügbare Optionen:

- `backend`: Holen Sie sich ein oder mehrere Storage Back-Ends von Astra Trident an.
- `snapshot`: Holen Sie sich einen oder mehrere Schnappschüsse von Astra Trident.
- `storageclass`: Holen Sie sich einen oder mehrere Storage-Kurse von Astra Trident.
- `volume`: Holen Sie sich ein oder mehrere Bände von Astra Trident.

volume Flaggen:

- * `-h, --help`: Hilfe für Volumen.
- * `--parentOfSubordinate string`: Abfrage auf untergeordnetes Quellvolumen begrenzen.
- * `--subordinateOf string`: Abfrage auf Untergebene beschränken.

images

Sie können die ausführen `images` Flagge, um eine Tabelle der Container-Bilder zu drucken, die Astra Trident benötigt.

```
Usage:
  tridentctl images [flags]
```

Markierungen:

- * `-h, --help``: Help for images.
- * `-V, --k8s-Version string``: Semantische Version des Kubernetes Clusters.

import volume

Sie können die ausführen `import volume` Befehl zum Importieren eines vorhandenen Volumes zu Astra Trident

```
Usage:
  tridentctl import volume <backendName> <volumeName> [flags]
```

Aliase:

`volume, v`

Markierungen:

- ``-f, --filename string`: Pfad zu YAML oder JSON PVC-Datei.
- ``-h, --help`: Hilfe für Lautstärke.
- ``--no-manage`: Nur PV/PVC erstellen. Nehmen Sie kein Lifecycle Management für Volumes an.

install

Sie können die ausführen `install` Flags für die Installation von Astra Trident.

```
Usage:
  tridentctl install [flags]
```

Markierungen:

- `--autosupport-image string`: Das Container-Image für AutoSupport Telemetrie (Standard „netapp/Trident AutoSupport:<current-version>“).
- `--autosupport-proxy string`: Die Adresse/der Port eines Proxy für den Versand von AutoSupport Telemetrie.
- `--enable-node-prep`: Versuch, benötigte Pakete auf Knoten zu installieren.
- `--generate-custom-yaml`: Erzeugen von YAML-Dateien ohne Installation von irgendetwas.
- `-h, --help`: Hilfe zur Installation.
- `--http-request-timeout`: Das HTTP-Anforderungs-Timeout für die REST-API des Trident-Controllers überschreiben (Standard 1m30s).
- `--image-registry string`: Die Adresse/der Port einer internen Bilddatenbank.
- `--k8s-timeout duration`: Die Zeitüberschreitung für alle Kubernetes-Operationen (Standard 3m0s).
- `--kubelet-dir string`: Der Host-Standort des internen Status von kubelet (Standard "/var/lib/kubelet").
- `--log-format string`: Das Astra Trident Logging-Format (Text, json) (Standard "Text").
- `--pv string`: Der Name des alten PV, das von Astra Trident verwendet wird, stellt sicher, dass dies nicht existiert (Standard "Dreizack").
- `--pvc string`: Der Name des alten PVC verwendet von Astra Trident, stellt sicher, dass dies nicht existiert (Standard "Dreizack").
- `--silence-autosupport`: AutoSupport Bundles nicht automatisch an NetApp senden (standardmäßig wahr).
- `--silent`: Während der Installation die meiste Leistung deaktivieren.
- `--trident-image string`: Das zu installierende Astra Trident-Image.
- `--use-custom-yaml`: Verwenden Sie alle bestehenden YAML-Dateien, die im Setup-Verzeichnis vorhanden sind.
- `--use-ipv6`: Nutzen Sie IPv6 für die Kommunikation von Astra Trident.

logs

Sie können die ausführen `logs` Flags zum Drucken der Protokolle von Astra Trident.

```
Usage:
  tridentctl logs [flags]
```

Markierungen:

- ``-a, --archive`: Erstellen Sie ein Stützarchiv mit allen Protokollen, sofern nicht anders angegeben.
- ``-h, --help`: Hilfe für Protokolle.
- ``-l, --log string`: Astra Trident Log to Display. Einer der Dreizack-Automatik-Operator ganz (Standard „Auto“).
- ``--node string`: Der Kubernetes-Knotenname, aus dem Node-Pod-Protokolle erfasst werden.
- ``-p, --previous`: Holen Sie sich die Protokolle für die frühere Container-Instanz, wenn sie existiert.
- ``--sidecars`: Holen Sie sich die Protokolle für die Sidecar-Container.

send

Sie können die ausführen `send` Befehl zum Senden einer Ressource vom Astra Trident.

```
Usage:
  tridentctl send [option]
```

Verfügbare Option:

`autosupport`: Senden Sie ein AutoSupport-Archiv an NetApp.

uninstall

Sie können die ausführen `uninstall` Flags zum Deinstallieren von Astra Trident.

```
Usage:
  tridentctl uninstall [flags]
```

Markierungen:

- * `-h, --help`: Hilfe zur Deinstallation.
- * `--silent`: Deaktivieren der meisten Ausgabe während der Deinstallation.

update

Sie können die ausführen `update` Befehle zum Ändern einer Ressource in Astra Trident.

```
Usage:
  tridentctl update [option]
```

Verfügbare Optionen:

`backend`: Aktualisieren Sie ein Backend im Astra Trident.

`version`

Sie können die ausführen `version` Flags zum Drucken der Version von `tridentctl` Und den Running Trident Service.

```
Usage:
  tridentctl version [flags]
```

Markierungen:

* `--client`: Nur Client-Version (kein Server erforderlich).

* `-h, --help`: Hilfe zur Version.

Pod Security Standards (PSS) und Security Context Constraints (SCC)

Kubernetes Pod Security Standards (PSS) und Pod Security Policies (PSP) definieren Berechtigungsebenen und schränken das Verhalten von Pods ein. OpenShift Security Context Constraints (SCC) definieren ebenfalls die Pod-Einschränkung speziell für die OpenShift Kubernetes Engine. Zur Bereitstellung dieser Anpassung ermöglicht Astra Trident während der Installation bestimmte Berechtigungen. In den folgenden Abschnitten werden die Berechtigungen von Astra Trident erläutert.



PSS ersetzt Pod Security Policies (PSP). PSP war in Kubernetes v1.21 veraltet und wird in v1.25 entfernt. Weitere Informationen finden Sie unter "[Kubernetes: Sicherheit](#)".

Erforderlicher Kubernetes-Sicherheitskontext und zugehörige Felder

| Berechtigung | Beschreibung |
|---------------|---|
| Privileged | Bei CSI müssen Mount-Punkte bidirektional sein. Das Trident Node-POD muss einen privilegierten Container ausführen. Weitere Informationen finden Sie unter " Kubernetes: Mount-Ausbreitung ". |
| Host-Netzwerk | Für den iSCSI-Daemon erforderlich. <code>iscsiadm</code> Managt iSCSI-Mounts und verwendet Host-Netzwerke für die Kommunikation mit dem iSCSI-Daemon. |
| Host-IPC | NFS nutzt Prozesskommunikation (IPC) mit dem NFSD. |

| Berechtigung | Beschreibung |
|--------------|---|
| Host-PID | Muss gestartet werden <code>rpc-statd</code> Für NFS. Astra Trident fragt die Host-Prozesse ab, um festzustellen, ob <code>rpc-statd</code> Wird vor dem Mounten von NFS-Volumes ausgeführt. |
| Sorgen | Der <code>SYS_ADMIN</code> Diese Funktion wird als Teil der Standardfunktionen für privilegierte Container bereitgestellt. Docker legt beispielsweise die folgenden Funktionen für privilegierte Container fest: <code>CapPrm: 0000003fffffffffff</code> <code>CapEff: 0000003fffffffffff</code> |
| Abt | Seccomp-Profil ist in privilegierten Containern immer „unbegrenzt“; daher kann es in Astra Trident nicht aktiviert werden. |
| SELinux | Auf OpenShift werden privilegierte Container im betrieben <code>spc_t</code> („Super Privileged Container“)-Domain, und unprivilegierte Container werden im ausgeführt <code>container_t</code> Domäne. Ein <code>containerd</code> , Mit <code>container-selinux</code> Installiert, alle Container werden im ausgeführt <code>spc_t</code> Domain, die SELinux effektiv deaktiviert. Aus diesem Grund wird Astra Trident nicht hinzugefügt <code>seLinuxOptions</code> Zusammen mit Containern. |
| DAC | Privilegierte Container müssen als Root ausgeführt werden. Nicht privilegierte Container werden als Root ausgeführt, um auf unix-Sockets zuzugreifen, die von CSI benötigt werden. |

Pod-Sicherheitsstandards (PSS)

| Etikett | Beschreibung | Standard |
|---|--|--|
| <code>pod-security.kubernetes.io/enforce</code> | Ermöglicht die Aufnahme der Trident Controller und Knoten im Namespace für die Installation. | <code>enforce: privileged</code> |
| <code>pod-security.kubernetes.io/enforce-version</code> | Ändern Sie nicht die Namespace-Bezeichnung. | <code>enforce-version: <version of the current cluster or highest version of PSS tested.></code> |



Das Ändern der Namespace-Labels kann dazu führen, dass Pods nicht geplant werden, ein „Error Creating: ...“ oder „Warnung: trident-csi-...“. Wenn dies geschieht, prüfen Sie, ob die Namespace-Bezeichnung für verwendet wird `privileged` Wurde geändert. Falls ja, installieren Sie Trident neu.

Pod-Sicherheitsrichtlinien (PSP)

| Feld | Beschreibung | Standard |
|--|---|-----------------------|
| <code>allowPrivilegeEscalation</code> | Privilegierte Container müssen die Eskalation von Berechtigungen ermöglichen. | <code>true</code> |
| <code>allowedCSIDrivers</code> | Trident verwendet keine kurzlebigen CSI-Inline-Volumes. | Leer |
| <code>allowedCapabilities</code> | Für Trident Container ohne Privilegien sind nicht mehr Funktionen erforderlich als für die Standardwerte. Privilegierte Container erhalten alle möglichen Funktionen. | Leer |
| <code>allowedFlexVolumes</code> | Trident verwendet kein a "FlexVolume-Treiber", Sie sind daher nicht in die Liste der zulässigen Volumen. | Leer |
| <code>allowedHostPaths</code> | Der Trident-Node-Pod hängt das Root-Dateisystem des Node zusammen, daher bietet es keinen Vorteil, diese Liste zu setzen. | Leer |
| <code>allowedProcMountTypes</code> | Trident verwendet keine <code>ProcMountTypes</code> . | Leer |
| <code>allowedUnsafeSysctls</code> | Trident erfordert keine Unsicherheit <code>sysctls</code> . | Leer |
| <code>defaultAddCapabilities</code> | Zu privilegierten Containern müssen keine Funktionen hinzugefügt werden. | Leer |
| <code>defaultAllowPrivilegeEscalation</code> | In jedem Trident Pod werden Berechtigungen erteilt. | <code>false</code> |
| <code>forbiddenSysctls</code> | Nein <code>sysctls</code> Zulässig. | Leer |
| <code>fsGroup</code> | Trident Container werden als Root ausgeführt. | <code>RunAsAny</code> |
| <code>hostIPC</code> | Das Mounten von NFS-Volumes erfordert die Kommunikation zwischen dem Host IPC und dem <code>nfsd</code> | <code>true</code> |
| <code>hostNetwork</code> | <code>iscsiadm</code> erfordert, dass das Hostnetzwerk mit dem iSCSI-Daemon kommunizieren kann. | <code>true</code> |
| <code>hostPID</code> | Host PID ist erforderlich, um zu überprüfen, ob <code>rpc-statd</code> Wird auf dem Node ausgeführt. | <code>true</code> |
| <code>hostPorts</code> | Trident verwendet keine Host Ports. | Leer |

| Feld | Beschreibung | Standard |
|--------------------------|---|-------------------------------|
| privileged | Trident Node-Pods müssen einen privilegierten Container ausführen, um Volumes mounten zu können. | true |
| readOnlyRootFilesystem | Trident Node-Pods müssen in das Node-Dateisystem schreiben. | false |
| requiredDropCapabilities | Trident Node-Pods führen einen privilegierten Container aus und können Funktionen nicht ablegen. | none |
| runAsGroup | Trident Container werden als Root ausgeführt. | RunAsAny |
| runAsUser | Trident Container werden als Root ausgeführt. | runAsAny |
| runtimeClass | Trident wird nicht verwendet RuntimeClasses. | Leer |
| seLinux | Trident ist nicht eingerichtet seLinuxOptions Weil es derzeit Unterschiede hinsichtlich der Handhabung von Container-Laufzeiten und Kubernetes-Distributionen für SELinux gibt. | Leer |
| supplementalGroups | Trident Container werden als Root ausgeführt. | RunAsAny |
| volumes | Trident Pods erfordern diese Volume-Plug-ins. | hostPath, projected, emptyDir |

Sicherheitskontexteinschränkungen (SCC)

| Etiketten | Beschreibung | Standard |
|--------------------------|---|----------|
| allowHostDirVolumePlugin | Trident-Node-Pods mounten das Root-Dateisystem des Node. | true |
| allowHostIPC | Das Mounten von NFS-Volumes erfordert die Kommunikation zwischen dem Host IPC und dem nfsd. | true |
| allowHostNetwork | Isctadm erfordert, dass das Hostnetzwerk mit dem iSCSI-Daemon kommunizieren kann. | true |
| allowHostPID | Host PID ist erforderlich, um zu überprüfen, ob rpc-statd Wird auf dem Node ausgeführt. | true |
| allowHostPorts | Trident verwendet keine Host Ports. | false |

| Etiketten | Beschreibung | Standard |
|--------------------------|--|-----------------|
| allowPrivilegeEscalation | Privilegierte Container müssen die Eskalation von Berechtigungen ermöglichen. | true |
| allowPrivilegedContainer | Trident Node-Pods müssen einen privilegierten Container ausführen, um Volumes mounten zu können. | true |
| allowedUnsafeSysctls | Trident erfordert keine Unsicherheit sysctls. | none |
| allowedCapabilities | Für Trident Container ohne Privilegien sind nicht mehr Funktionen erforderlich als für die Standardwerte. Privilegierte Container erhalten alle möglichen Funktionen. | Leer |
| defaultAddCapabilities | Zu privilegierten Containern müssen keine Funktionen hinzugefügt werden. | Leer |
| fsGroup | Trident Container werden als Root ausgeführt. | RunAsAny |
| groups | Dieses SCC ist speziell für Trident bestimmt und an den Anwender gebunden. | Leer |
| readOnlyRootFilesystem | Trident Node-Pods müssen in das Node-Dateisystem schreiben. | false |
| requiredDropCapabilities | Trident Node-Pods führen einen privilegierten Container aus und können Funktionen nicht ablegen. | none |
| runAsUser | Trident Container werden als Root ausgeführt. | RunAsAny |
| seLinuxContext | Trident ist nicht eingerichtet <code>seLinuxOptions</code> Weil es derzeit Unterschiede hinsichtlich der Handhabung von Container-Laufzeiten und Kubernetes-Distributionen für SELinux gibt. | Leer |
| seccompProfiles | Privilegierte Container laufen immer „unbegrenzt“. | Leer |
| supplementalGroups | Trident Container werden als Root ausgeführt. | RunAsAny |
| users | Es ist ein Eintrag verfügbar, um diesen SCC an den Trident-Benutzer im Trident Namespace zu binden. | k. A. |

| Etiketten | Beschreibung | Standard |
|------------------|---|--|
| volumes | Trident Pods erfordern diese Volume-Plug-ins. | hostPath, downwardAPI, projected, emptyDir |

Rechtliche Hinweise

Rechtliche Hinweise ermöglichen den Zugriff auf Copyright-Erklärungen, Marken, Patente und mehr.

Urheberrecht

["https://www.netapp.com/company/legal/copyright/"](https://www.netapp.com/company/legal/copyright/)

Marken

NetApp, das NETAPP Logo und die auf der NetApp Markenseite aufgeführten Marken sind Marken von NetApp Inc. Andere Firmen- und Produktnamen können Marken der jeweiligen Eigentümer sein.

["https://www.netapp.com/company/legal/trademarks/"](https://www.netapp.com/company/legal/trademarks/)

Patente

Eine aktuelle Liste der NetApp Patente finden Sie unter:

<https://www.netapp.com/pdf.html?item=/media/11887-patentspage.pdf>

Datenschutzrichtlinie

["https://www.netapp.com/company/legal/privacy-policy/"](https://www.netapp.com/company/legal/privacy-policy/)

Open Source

Sie können das Copyright und die Lizenzen, die in der NetApp Software für Astra Trident verwendet werden, auch in der Notizen für die jeweilige Version unter lesen <https://github.com/NetApp/trident/>.

Copyright-Informationen

Copyright © 2024 NetApp. Alle Rechte vorbehalten. Gedruckt in den USA. Dieses urheberrechtlich geschützte Dokument darf ohne die vorherige schriftliche Genehmigung des Urheberrechtinhabers in keiner Form und durch keine Mittel – weder grafische noch elektronische oder mechanische, einschließlich Fotokopieren, Aufnehmen oder Speichern in einem elektronischen Abrufsystem – auch nicht in Teilen, vervielfältigt werden.

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

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

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

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

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

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

Markeninformationen

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