Astra Trident 23.01.1 Dokumentation

Astra Trident

NetApp
July 17, 2025

This PDF was generated from https://docs.netapp.com/de-de/trident-2301/index.html on July 17, 2025.
Always check docs.netapp.com for the latest.

Inhalt

Astra Trident 23.01.1 Dokumentation . . . 1
Versionshinweise . . . 2
Was ist neu in 23.01.1 . . . 2
Korrekturen . . . 2
Änderungen in 23.01 . . . 2
Korrekturen . . . 2
Vorgestellt Werden . . . 2
Abschreibungen . . . 3
Änderungen in 22.10 . . . 3
Korrekturen . . . 3
Vorgestellt Werden . . . 4
Abschreibungen . . . 4
Änderungen in 22.07 . . . 4
Korrekturen . . . 4
Vorgestellt Werden . . . 4
Abschreibungen . . . 5
Umzüge . . . 5
Dokumentation . . . 5
Änderungen in 22.04 . . . 5
Korrekturen . . . 5
Vorgestellt Werden . . . 6
Umzüge . . . 6
Änderungen in 22.01.1 . . . 6
Korrekturen . . . 6
Änderungen in 22.01.0 . . . 6
Korrekturen . . . 6
Vorgestellt Werden . . . 6
Abschreibungen . . . 7
Änderungen in 21.10.1 . . . 7
Korrekturen . . . 7
Änderungen in 21.10.0 . . . 7
Korrekturen . . . 7
Vorgestellt Werden . . . 8
Experimentelle Verbesserungen . . . 8
Bekannte Probleme . . . 8
Weitere Informationen . . . 9
Konzepte . . . 10
Erfahren Sie mehr über Astra Trident . . . 10
Überblick . . . 10
Unterstützte Kubernetes-Cluster-Architekturen . . . 10
Was ist Astra? . . . 10
Finden Sie weitere Informationen . . . 11
ONTAP-Treiber . . . 11

Erfahren Sie mehr über die ONTAP Storage-Treiber 11
Bereitstellung 12
Storage-Klassen-Zuordnung 12
Volume-Erstellung 13
Volume Snapshots 13
Erfahren Sie mehr über die Erstellung von Volume Snapshots 13
Virtuelle Pools 14
Erfahren Sie mehr über virtuelle Pools 14
Volume-Zugriffsgruppen 15
Erfahren Sie mehr über Volume Access Groups 15
Los geht’s 17
Probieren Sie es aus 17
Erfahren Sie mehr über die Testversion 17
Anforderungen 17
Entscheidende Informationen zu Astra Trident 23.01 17
Unterstützte Frontends (Orchestrators) 17
Unterstützte Back-Ends (Storage) 18
Anforderungen an die Funktionen 18
Getestete Host-Betriebssysteme 19
Host-Konfiguration 19
Konfiguration des Storage-Systems 19
Astra Trident-Ports 19
Container-Images und entsprechende Kubernetes-Versionen 19
Installation Von Astra Trident 22
Erfahren Sie mehr über die Installation von Astra Trident 22
Installation über den Trident Operator 26
Installieren Sie mit tridentctl 52
Was kommt als Nächstes? 56
Schritt 1: Erstellen Sie ein Backend 56
Schritt 2: Erstellen Sie eine Storage-Klasse 57
Schritt 3: Stellen Sie Ihr erstes Volumen bereit 59
Schritt 4: Mounten Sie die Volumes in einem POD 60
Managen Sie Astra Trident 62
Upgrade Astra Trident 62
Upgrade Astra Trident 62
Upgrade mit dem Bediener 63
Upgrade mit tridentctl 71
Deinstallieren Sie Astra Trident 75
Deinstallieren Sie mit Helm 75
Deinstallieren Sie die Deinstallation mit dem Trident-Operator 76
Deinstallieren Sie mit `tridentctl` 77
Downgrade Astra Trident 77
Wenn sie heruntergestuft werden müssen 77
Wenn Sie nicht herunterstufen 77
Downgrade bei der Installation von Astra Trident mit dem Operator 78

Downgrade bei Installation von Astra Trident mit `tridentctl` 79
Nutzen Sie Astra Trident 82
Bereiten Sie den Knoten „Worker“ vor 82
Auswahl der richtigen Werkzeuge 82
Ermittlung des Node-Service 82
NFS Volumes 83
ISCSI-Volumes 83
Back-Ends konfigurieren 86
Azure NetApp Dateien 87
Cloud Volumes Service für Google Cloud-Back-End konfigurieren 99
Konfigurieren Sie ein NetApp HCI- oder SolidFire-Backend 115
Konfigurieren Sie ein Backend mit ONTAP-SAN-Treibern 122
Konfigurieren Sie ein ONTAP-NAS-Back-End 144
Amazon FSX für NetApp ONTAP 170
Back-Ends mit kubectl erstellen 182
`TridentBackendConfig` 182
Schritte im Überblick 183
Schritt: Ein Kubernetes Secret erstellen 184
Schritt 2: Erstellen Sie die `TridentBackendConfig` CR 185
Schritt 3: Überprüfen Sie den Status des `TridentBackendConfig` CR 186
(Optional) Schritt 4: Weitere Informationen 187
Führen Sie das Back-End-Management mit kubectl durch 189
Löschen Sie ein Back-End 189
Zeigen Sie die vorhandenen Back-Ends an 189
Aktualisieren Sie ein Backend 189
Back-End-Management mit tridentctl 190
Erstellen Sie ein Backend 190
Löschen Sie ein Back-End 190
Zeigen Sie die vorhandenen Back-Ends an 191
Aktualisieren Sie ein Backend 191
Identifizieren Sie die Storage-Klassen, die ein Backend nutzen 191
Wechseln Sie zwischen den Back-End-Managementoptionen 192
Managen `tridentctl` Back-Ends mit `TridentBackendConfig` 192
Managen `TridentBackendConfig` Back-Ends mit `tridentctl` 196
Management von Storage-Klassen 198
Entwurf einer Storage-Klasse 198
Erstellen Sie eine Speicherklasse 198
Löschen Sie eine Speicherklasse 199
Sehen Sie sich die vorhandenen Speicherklassen an 199
Legen Sie eine Standardspeicherklasse fest 199
Das Backend für eine Storage-Klasse ermitteln 200
Durchführung von Volume-Vorgängen 200
Verwenden Sie die CSI-Topologie 200
Arbeiten Sie mit Snapshots 208

Erweitern Sie Volumes . . . 212
Volumes importieren . . . 219
Ein NFS-Volume kann über Namespaces hinweg genutzt werden . . . 225
Funktionen . . . 225
Schnellstart . . . 226
Konfigurieren Sie die Namensräume für Quelle und Ziel . . . 227
Löschen eines freigegebenen Volumes . . . 228
Nutzung `tridentctl get` Zum Abfragen von untergeordneten Volumes . . . 228
Einschränkungen . . . 229
Finden Sie weitere Informationen . . . 229
Überwachen Sie Astra Trident . . . 229
Schritt 1: Definieren Sie ein Prometheus-Ziel . . . 230
Schritt: Erstellen Sie einen Prometheus ServiceMonitor . . . 230
Schritt 3: Abfrage der Trident-Kennzahlen mit PromQL . . . 231
AutoSupport Telemetrie von Astra Trident mit Thema . . . 232
Deaktivieren Sie Astra Trident Metriken . . . 233
Astra Trident für Docker . . . 234
Voraussetzungen für die Bereitstellung . . . 234
Überprüfen Sie die Anforderungen . . . 234
Implementieren Sie Astra Trident . . . 237
Docker Managed Plug-in-Methode (Version 1.13/17.03 und höher) . . . 237
Herkömmliche Methode (Version 1.12 oder früher) . . . 239
Starten Sie Astra Trident beim Systemstart . . . 240
Astra Trident upgraden oder deinstallieren . . . 241
Upgrade . . . 242
Deinstallieren . . . 243
Arbeiten mit Volumes . . . 243
Erstellen eines Volumes . . . 243
Entfernen Sie ein Volume . . . 244
Klonen Sie ein Volume . . . 244
Zugriff auf extern erstellte Volumes . . . 246
Treiberspezifische Volume-Optionen . . . 246
Sammelt Protokolle . . . 252
Sammelt Protokolle für die Fehlerbehebung . . . 252
Allgemeine Tipps zur Fehlerbehebung . . . 252
Management mehrerer Astra Trident Instanzen . . . 253
Schritte für Docker Managed Plug-in (Version 1.13/17.03 oder höher) . . . 253
Schritte für herkömmliche (Version 1.12 oder früher) . . . 253
Optionen für die Storage-Konfiguration . . . 254
Globale Konfigurationsoptionen . . . 254
ONTAP-Konfiguration . . . 255
Konfiguration von Element Software . . . 261
Bekannte Probleme und Einschränkungen . . . 263
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. . . . 263

Volume-Namen müssen mindestens 2 Zeichen lang sein. . . . 264
Docker Swarm hat bestimmte Verhaltensweisen, die Astra Trident nicht durch jede Storage- und Treiberkombination unterstützen können. . . . 264
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. . . . 264
Häufig gestellte Fragen . . . 265
Allgemeine Fragen . . . 265
Wie oft wird Astra Trident veröffentlicht? . . . 265
Unterstützt Astra Trident alle Funktionen, die in einer bestimmten Version von Kubernetes verfügbar sind? . . . 265
Verfügt Astra Trident über irgendwelche Abhängigkeiten von anderen NetApp Produkten für seine Funktionsweise? . . . 265
Wie erhalte ich vollständige Astra Trident Konfigurationsdetails? . . . 265
Kann ich Metriken abrufen, wie Storage von Astra Trident bereitgestellt wird? . . . 265
Ändert sich die Benutzererfahrung, wenn Astra Trident als CSI-Bereitstellung verwendet wird? . . . 265
Installation und Verwendung von Astra Trident in einem Kubernetes Cluster . . . 265
Welche Versionen werden von unterstützt `etcd`? . . . 266
Unterstützt Astra Trident eine Offline-Installation von einer privaten Registry? . . . 266
Kann Astra Trident Remote installiert werden? . . . 266
Kann ich Hochverfügbarkeit mit Astra Trident konfigurieren? . . . 266
Benötigt Astra Trident Zugriff auf den kube-System-Namespace? . . . 266
Welche Rollen und Privilegien werden von Astra Trident verwendet? . . . 266
Kann ich lokal die genauen Manifest-Dateien generieren, die Astra Trident zur Installation verwendet? 266
Kann ich dieselbe ONTAP Backend-SVM für zwei separate Astra Trident Instanzen für zwei separate Kubernetes Cluster nutzen? . . . 266
Ist es möglich, Astra Trident unter ContainerLinux (früher CoreOS) zu installieren? . . . 266
Kann ich Astra Trident mit NetApp Cloud Volumes ONTAP verwenden? . . . 267
Funktioniert Astra Trident mit Cloud Volumes Services? . . . 267
Fehlerbehebung und Support . . . 267
Bietet NetApp Unterstützung für Astra Trident? . . . 267
Wie kann ich einen Support-Fall anheben? . . . 267
Wie generiere ich ein Support Log-Paket? . . . 267
Was muss ich tun, wenn ich einen Antrag auf eine neue Funktion stellen muss? . . . 267
Wo kann ich einen Defekt aufwerfen? . . . 267
Was passiert, wenn ich schnell Fragen zu Astra Trident habe, die ich klären muss? Gibt es eine Gemeinschaft oder ein Forum? . . . 267
Das Passwort meines Storage-Systems hat sich geändert und Astra Trident funktioniert nicht mehr. Wie kann ich es wiederherstellen? . . . 267
Astra Trident kann meinen Kubernetes-Node nicht finden. Wie kann ich das beheben? . . . 268
Geht der Trident Pod verloren, gehen die Daten verloren? . . . 268
Upgrade Astra Trident . . . 268
Kann ich ein Upgrade von einer älteren Version direkt auf eine neuere Version durchführen (einige Versionen werden übersprungen)? . . . 268
Ist es möglich, Trident auf eine vorherige Version herunterzustufen? . . . 268

Back-Ends und Volumes managen . . . 268
Muss ich Management- und Daten-LIFs in einer ONTAP-Back-End-Definitionsdatei definieren? . . . 268
Kann Astra Trident CHAP für ONTAP-Back-Ends konfigurieren? . . . 268
Wie schaffe ich Exportrichtlinien mit Astra Trident? . . . 269
Können wir einen Port im DataLIF angeben? . . . 269
Können IPv6-Adressen für das Management und die Daten-LIFs verwendet werden? . . . 269
Ist es möglich, die Management LIF auf dem Backend zu aktualisieren? . . . 269
Ist es möglich, die Daten-LIF auf dem Backend zu aktualisieren? . . . 269
Kann ich in Astra Trident mehrere Back-Ends für Kubernetes erstellen? . . . 269
Wie speichert Astra Trident Back-End-Anmeldedaten? . . . 269
Wie wählt Astra Trident ein spezifisches Backend aus? . . . 269
Wie kann ich sicherstellen, dass Astra Trident nicht über ein spezifisches Backend bereitgestellt wird? 270
Wenn es mehrere Back-Ends derselben Art gibt, wie wählt Astra Trident das zu verwendende Back-End aus? . . . 270
Unterstützt Astra Trident bidirektionales CHAP mit Element/SolidFire? . . . 270
Wie implementiert Astra Trident qtrees auf einem ONTAP Volume? Wie viele qtrees können auf einem einzelnen Volume implementiert werden? . . . 270
Wie kann ich Unix Berechtigungen für Volumes festlegen, die auf ONTAP NAS bereitgestellt werden? 270
Wie kann ich bei der Bereitstellung eines Volumes einen expliziten Satz von ONTAP-NFS-Mount-Optionen konfigurieren? . . . 270
Wie lege ich die bereitgestellten Volumes auf eine bestimmte Exportrichtlinie fest? . . . 270
Wie setze ich mit ONTAP die Volume-Verschlüsselung durch Astra Trident ein? . . . 270
Wie implementiert man QoS für ONTAP am besten über Astra Trident? . . . 271
Wie soll ich über Astra Trident Thin oder Thick Provisioning angeben? . . . 271
Wie kann ich sicherstellen, dass die verwendeten Volumes nicht gelöscht werden, auch wenn ich aus Versehen die PVC lösche? . . . 271
Kann ich die von Astra Trident erstellten NFS PVCs ausbauen? . . . 271
Kann ich es in Astra Trident importieren, wenn ich ein Volume habe, das außerhalb von Astra Trident erstellt wurde? . . . 271
Kann ich ein Volume importieren, während es sich in SnapMirror Data Protection (DP) oder offline Modus befindet? . . . 271
Kann ich die von Astra Trident erstellten iSCSI PVCs erweitern? . . . 271
Wie wird ein Ressourcenkontingent auf ein NetApp Cluster übersetzt? . . . 271
Kann ich mit Astra Trident Volume Snapshots erstellen? . . . 272
Welche Faktoren sind die Faktoren, die die Volume-Snapshots von Astra Trident unterstützen? . . . 272
Wie kann ich ein Snapshot-Backup eines von Astra Trident bereitgestellten Volumes mit ONTAP erstellen? . . . 272
Kann ich einen prozentualen Anteil der Snapshot-Reserve für ein über Astra Trident bereitgestelltes Volume festlegen? . . . 272
Kann ich direkt auf das Snapshot-Verzeichnis des Volumes zugreifen und Dateien kopieren? . . . 272
Kann ich SnapMirror für Volumes über Astra Trident einrichten? . . . 272
Wie kann ich persistente Volumes auf einen bestimmten ONTAP Snapshot wiederherstellen? . . . 273
Kann Trident Volumes auf SVMs bereitstellen, die ein Load Sharing Mirror konfiguriert haben? . . . 273
Wie lässt sich die Storage-Klassennutzung für jeden Kunden/Mandanten trennen? . . . 273
Unterstützung . . . 274

Fehlerbehebung . . . 275
Allgemeine Fehlerbehebung . . . 275
Fehlerbehebung bei einer nicht erfolgreichen Trident-Implementierung mithilfe des Betreibers . . . 277
Fehlerbehebung bei einer nicht erfolgreichen Trident-Implementierung mit `tridentctl` . . . 279
Best Practices und Empfehlungen . . . 280
Einsatz . . . 280
Implementieren Sie diesen in einem dedizierten Namespace . . . 280
Verwenden Sie Kontingente und Bereichsgrenzen, um den Storage-Verbrauch zu kontrollieren . . . 280
Storage-Konfiguration . . . 280
Plattformübersicht . . . 280
Best Practices für ONTAP und Cloud Volumes ONTAP . . . 281
SolidFire Best Practices in sich vereint . . . 285
Wo finden Sie weitere Informationen? . . . 287
Integration Von Astra Trident . . . 287
Auswahl und Implementierung der Treiber . . . 288
Design der Storage-Klasse . . . 291
Virtual Pool Design . . . 292
Volume-Vorgänge . . . 293
OpenShift Services implementieren . . . 295
Kennzahlungsservice . . . 297
Datensicherung . . . 298
Sichern Sie die `etcd` Cluster-Daten . . . 298
Daten mit ONTAP Snapshots wiederherstellen . . . 299
Datenreplizierung mit ONTAP . . . 300
Daten mit Element Snapshots wiederherstellen . . . 303
Sicherheit . . . 303
Sicherheit . . . 303
Linux Unified Key Setup (LUKS) . . . 305
Referenz . . . 311
Astra Trident-Ports . . . 311
Astra Trident-Ports . . . 311
Astra Trident REST-API . . . 311
Wann die REST-API verwendet werden soll . . . 311
REST-API wird verwendet . . . 311
Befehlszeilenoptionen . . . 312
Protokollierung . . . 312
Kubernetes . . . 312
Docker . . . 313
RUHE . . . 313
NetApp Produkte sind in Kubernetes integriert . . . 313
Astra . . . 313
ONTAP . . . 313
Cloud Volumes ONTAP . . . 314
Amazon FSX für NetApp ONTAP . . . 314
Element Software . . . 314

NetApp HCI . . . 314
Azure NetApp Dateien . . . 314
Cloud Volumes Service für Google Cloud . . . 314
Kubernetes und Trident Objekte . . . 314
Wie interagieren die Objekte miteinander? . . . 315
Kubernetes `PersistentVolumeClaim` Objekte . . . 315
Kubernetes `PersistentVolume` Objekte . . . 317
Kubernetes `StorageClass` Objekte . . . 317
Kubernetes `VolumeSnapshotClass` Objekte . . . 321
Kubernetes `VolumeSnapshot` Objekte . . . 322
Kubernetes `VolumeSnapshotContent` Objekte . . . 322
Kubernetes `CustomResourceDefinition` Objekte . . . 322
Trident `StorageClass` Objekte . . . 323
Trident Back-End-Objekte . . . 323
Trident `StoragePool` Objekte . . . 324
Trident `Volume` Objekte . . . 324
Trident `Snapshot` Objekte . . . 325
Astra Trident `ResourceQuota` Objekt . . . 326
Tridentctl-Befehle und -Optionen . . . 327
Verfügbare Befehle und Optionen . . . 327
`create` . . . 328
`delete` . . . 329
`get` . . . 329
`images` . . . 329
`import volume` . . . 330
`install` . . . 330
`logs` . . . 331
`send` . . . 331
`uninstall` . . . 331
`update` . . . 332
`upgrade` . . . 332
`version` . . . 332
Pod Security Standards (PSS) und Security Context Constraints (SCC) . . . 332
Erforderlicher Kubernetes-Sicherheitskontext und zugehörige Felder . . . 333
Pod-Sicherheitsstandards (PSS) . . . 334
Pod-Sicherheitsrichtlinien (PSP) . . . 334
Sicherheitskontexteinschränkungen (SCC) . . . 336
Frühere Versionen der Dokumentation . . . 338
Rechtliche Hinweise . . . 339
Urheberrecht . . . 339
Marken . . . 339
Patente . . . 339

Datenschutzrichtlinie 339
Open Source 339

Astra Trident 23.01.1 Dokumentation

Versionshinweise

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.

Was ist neu 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 1.26 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 yaml 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-Umschalterkennung für ONTAP-Treiber ist jetzt allgemein verfügbar.

Abschreibungen

- **Kubernetes:** Aktualisiert unterstützt mindestens Kubernetes auf 1.20.
- Astra Data Store (ADS)-Treiber entfernt.
- Unterstützung für wurde entfernt `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 opportun auf das neue igroup-Schema umgeschaltet, wenn Trident feststellt, dass der Einsatz sicher ist, ohne dass aktive Workloads beeinträchtigt werden.
- Enthält eine ResourceQuota mit Trident-Installationen, um sicherzustellen, dass Trident DemonSet geplant ist, wenn die PriorityClass-Nutzung standardmäßig beschränkt ist.

- Unterstützung für Netzwerkfunktionen wurde dem ANF-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.
  - Verringert die Nodevorbereitung auf einfache Host-Info und aktive Serviceerkennung, um eine Bestätigung für den bestmöglichen Aufwand zu machen, dass NFS/iSCSI-Dienste auf Worker-Knoten 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-Speicherklassenparameter 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")
- Das Problem wurde behoben, bei dem der ANF-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.
- ANF-Treiber wurde behoben, um doppelte Ressourcennamen zu ermöglichen und Operationen auf einen einzelnen Speicherort zu beschränken.
- ANF-Treiber wurde behoben, um doppelte Ressourcennamen zu ermöglichen und Operationen 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 ANF-Volumes 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 Mindestrückstellunggröße 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"

Konzepte

Erfahren Sie mehr über Astra Trident

Astra Trident ist ein vollständig unterstütztes Open-Source-Projekt, das von NetApp im Rahmen der durchgeführt wird "Astra Produktfamilie". Es wurde entwickelt, damit Sie die Persistenzanforderungen Ihrer Container-Applikationen mithilfe von branchenüblichen Schnittstellen wie dem Container-Storage-Interface (CSI) erfüllen können.

Überblick

Astra Trident wird in Kubernetes Clustern als Pods implementiert und bietet dynamische Storage-Orchestrierungs-Services für Ihre Kubernetes-Workloads. Es ermöglicht Ihren Container-Applikationen die schnelle und einfache Nutzung von persistentem Storage aus dem breiten Portfolio von NetApp, das ONTAP (AFF/FAS/Select/Cloud/Amazon FSX für NetApp ONTAP), Element Software (NetApp HCI/SolidFire), Azure NetApp Files Service und Cloud Volumes Service auf Google Cloud umfasst.

Astra Trident ist außerdem eine grundlegende Technologie für den NetApp Astra. Er eignet sich für Ihre Datensicherung, Disaster Recovery, Portabilität und Migration von Kubernetes-Workloads und nutzt die branchenführende Datenmanagement-Technologie von NetApp für Snapshots, Backups, Replizierung und Klonen.

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.

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 mithilfe von Astra Trident bereit. Das bewährte und umfangreiche Storage-Portfolio von NetApp umfasst sowohl Public Clouds als auch On-Premises. Außerdem bietet es umfassende erweiterte, applikationsspezifische Datenmanagementfunktionen wie Snapshot, Backup und Wiederherstellung, Aktivitätsprotokolle und aktives Klonen für Datensicherung, Disaster/Daten-Recovery, Datenaudits und Migrationsanwendungsfälle für Kubernetes-Workloads.

Auf der Astra-Seite können Sie sich für eine kostenlose Testversion anmelden.

Finden Sie weitere Informationen

- "Die NetApp Astra-Produktfamilie"
- "Dokumentation des Astra Control Service"
- "Astra Control Center-Dokumentation"
- "Astra API-Dokumentation"

ONTAP-Treiber

Astra Trident bietet fünf einzigartige ONTAP-Storage-Treiber für die Kommunikation mit ONTAP Clustern. Erfahren Sie mehr darüber, wie jeder Treiber die Erstellung von Volumes und Zugriffssteuerung sowie seine Funktionen übernimmt.

Erfahren Sie mehr über die ONTAP Storage-Treiber

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.

- Sie müssen verwenden `ontap-nas` Für produktive Workloads, die Datensicherung, Disaster Recovery und Mobilität erfordern.
- Nutzung `ontap-san-economy` Nach einer voraussichtlichen Volume-Nutzung ist zu erwarten, dass sie wesentlich höher ist, als das von ONTAP unterstützt wird.
- Nutzung `ontap-nas-economy` Nur in dem eine zu erwartende Volume-Nutzung größer als die von ONTAP wird, und `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.

Treiber	Protokoll	VolumeModus	Unterstützte Zugriffsmodi	Unterstützte Filesysteme
`ontap-nas`	NFS	Dateisystem	RWO, ROX, RWX	„“, nfs
`ontap-nas-economy`	NFS	Dateisystem	RWO, ROX, RWX	„“, nfs
`ontap-nas-flexgroup`	NFS	Dateisystem	RWO, ROX, RWX	„“, nfs
`ontap-san`	ISCSI	Block-Storage	RWO, ROX, RWX	Kein Filesystem, rohes Block-Gerät

Treiber	Protokoll	VolumeModus	Unterstützte Zugriffsmodi	Unterstützte Filesysteme
`ontap-san`	ISCSI	Dateisystem	RWO, ROX RWX ist im Volume-Modus des Dateisystems nicht verfügbar.	`xfs`, `ext3`, `ext4`
`ontap-san-economy`	ISCSI	Block-Storage	RWO, ROX, RWX	Kein Filesystem, rohes Block-Gerät
`ontap-san-economy`	ISCSI	Dateisystem	RWO, ROX RWX ist im Volume-Modus des Dateisystems nicht verfügbar.	`xfs`, `ext3`, `ext4`

ONTAP-Back-Ends können mit Anmeldeinformationen für eine Sicherheitsrolle (Benutzername/Passwort) oder mit dem privaten Schlüssel und dem Zertifikat, das auf dem ONTAP-Cluster installiert ist, authentifiziert werden. Sie können vorhandene Back-Ends aktualisieren, um mit von einem Authentifizierungsmodus in den anderen zu verschieben `tridentctl update backend`.

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 den Storage-Klassen des ausstehenden Volume.

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 ab `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. Die Snapshot-Technologie von NetApp liefert höhere Stabilität, Skalierbarkeit, Wiederherstellbarkeit und Performance als vergleichbare Snapshot-Technologien. 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. Die Snapshot-Technologie von NetApp liefert höhere Stabilität, Skalierbarkeit, Wiederherstellbarkeit und Performance als vergleichbare Snapshot-Technologien. 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. Die FlexClone Technologie von ONTAP ermöglicht es sogar von den größten Datensätzen fast unmittelbar 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

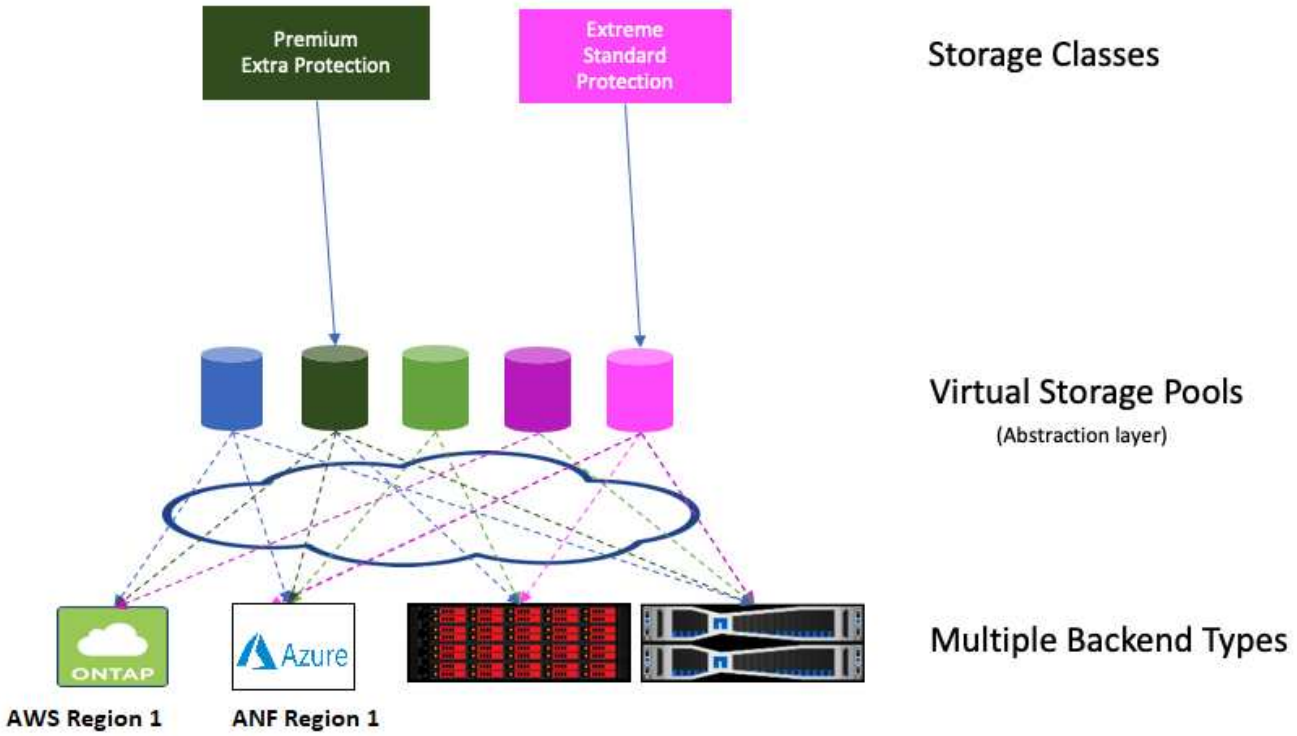
von PVS aus diesen Snapshots wird mithilfe der FlexClone Technologie für unterstützte ONTAP- und CVS-Back-Ends durchgeführt. Bei der Erstellung eines PV aus einem Snapshot ist das zugrunde liegende Volume ein FlexClone des übergeordneten Volume des Snapshots. Der `solidfire-san` Der Treiber verwendet 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.

Während Astra Trident neue Volumes mit den konfigurierten Zugriffsgruppen verknüpft, erstellt oder verwaltet sie nicht selbst Zugriffsgruppen. Die Zugriffsgruppe(n) muss vorhanden sein, bevor das Storage-Backend zum Astra Trident hinzugefügt wird. Sie müssen die iSCSI-IQNs von jedem Node im Kubernetes-Cluster enthalten, der die durch das Backend bereitgestellten Volumes potenziell 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 einem ändern, das den Standard verwendet `trident` Zugriffsgruppe für eine Gruppe, die auch andere verwendet, geben Sie die ID für die ein `trident` Zugriffsgruppe in der Liste.

Los geht's

Probieren Sie es aus

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

Erfahren Sie mehr über die Testversion

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

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

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

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

Anforderungen

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

Entscheidende Informationen zu Astra Trident 23.01

Sie müssen die folgenden wichtigen Informationen über Astra Trident lesen.

** Informationen über Astra TripereIT **

- Kubernetes 1.26 wird jetzt in Trident unterstützt. Upgrade von Trident vor dem Upgrade von Kubernetes.
- Astra Trident setzt die Verwendung von Multipathing-Konfiguration in SAN-Umgebungen strikt um und empfiehlt den Nutzen von `find_multipaths: no` In Multipath.conf Datei.

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

Unterstützte Frontends (Orchestrators)

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

- Anthos On-Prem (VMware) und Anthos auf Bare Metal 1.9, 1.10 und 1.11

- Kubernetes 1.21 - 1.26
- Mirantis Kubernetes Engine 3.5
- OpenShift 4.9 - 4.12

Der Trident-Operator wird durch folgende Versionen unterstützt:

- Anthos On-Prem (VMware) und Anthos auf Bare Metal 1.9, 1.10 und 1.11
- Kubernetes 1.21 - 1.26
- OpenShift 4.9 - 4.12

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-basierten Bedienerinstallation".

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?
CSI Trident	1.21 - 1.26	Nein
Volume Snapshots	1.21 - 1.26	Nein
PVC aus Volume Snapshots	1.21 - 1.26	Nein
ISCSI PV-Größe	1.21 - 1.26	Nein
Bidirektionales ONTAP-CHAP	1.21 - 1.26	Nein

Merkmal	Kubernetes-Version	Funktionstore erforderlich?
Dynamische Exportrichtlinien	1.21 - 1.26	Nein
Trident Operator	1.21 - 1.26	Nein
CSI-Topologie	1.21 - 1.26	Nein

Getestete Host-Betriebssysteme

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

- Versionen von redhat CoreOS (RHCOS) werden von OpenShift Container Platform unterstützt
- RHEL 8 ODER HÖHER
- Ubuntu 22.04 oder höher
- Windows Server 2019

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

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

Host-Konfiguration

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
V1.21,0	• docker.io/netapp/Trident:23.01.1 • docker.io/netapp/Trident-AutoSupport:23.01 • Registry.k8s.io/SIG-Storage/csi-provisioner:v3.4.0 • Registry.k8s.io/SIG-Storage/csi-Attacher:v4.1.0 • Registry.k8s.io/SIG-Storage/csi-resizer:v1.7.0 • Registry.k8s.io/SIG-Storage/csi-snapshotter:v6.2.1 • Registry.k8s.io/SIG-Storage/csi-Node-driver-Registrar:v2.7.0 • docker.io/netapp/Trident-Operator:23.01.1 (optional)
V1.22.0	• docker.io/netapp/Trident:23.01.1 • docker.io/netapp/Trident-AutoSupport:23.01 • Registry.k8s.io/SIG-Storage/csi-provisioner:v3.4.0 • Registry.k8s.io/SIG-Storage/csi-Attacher:v4.1.0 • Registry.k8s.io/SIG-Storage/csi-resizer:v1.7.0 • Registry.k8s.io/SIG-Storage/csi-snapshotter:v6.2.1 • Registry.k8s.io/SIG-Storage/csi-Node-driver-Registrar:v2.7.0 • docker.io/netapp/Trident-Operator:23.01.1 (optional)
V1.23.0	• docker.io/netapp/Trident:23.01.1 • docker.io/netapp/Trident-AutoSupport:23.01 • Registry.k8s.io/SIG-Storage/csi-provisioner:v3.4.0 • Registry.k8s.io/SIG-Storage/csi-Attacher:v4.1.0 • Registry.k8s.io/SIG-Storage/csi-resizer:v1.7.0 • Registry.k8s.io/SIG-Storage/csi-snapshotter:v6.2.1 • Registry.k8s.io/SIG-Storage/csi-Node-driver-Registrar:v2.7.0 • docker.io/netapp/Trident-Operator:23.01.1 (optional)

Kubernetes-Version	Container-Image
V1.24.0	• docker.io/netapp/Trident:23.01.1 • docker.io/netapp/Trident-AutoSupport:23.01 • Registry.k8s.io/SIG-Storage/csi-provisioner:v3.4.0 • Registry.k8s.io/SIG-Storage/csi-Attacher:v4.1.0 • Registry.k8s.io/SIG-Storage/csi-resizer:v1.7.0 • Registry.k8s.io/SIG-Storage/csi-snapshotter:v6.2.1 • Registry.k8s.io/SIG-Storage/csi-Node-driver-Registrar:v2.7.0 • docker.io/netapp/Trident-Operator:23.01.1 (optional)
V1.25.0	• docker.io/netapp/Trident:23.01.1 • docker.io/netapp/Trident-AutoSupport:23.01 • Registry.k8s.io/SIG-Storage/csi-provisioner:v3.4.0 • Registry.k8s.io/SIG-Storage/csi-Attacher:v4.1.0 • Registry.k8s.io/SIG-Storage/csi-resizer:v1.7.0 • Registry.k8s.io/SIG-Storage/csi-snapshotter:v6.2.1 • Registry.k8s.io/SIG-Storage/csi-Node-driver-Registrar:v2.7.0 • docker.io/netapp/Trident-Operator:23.01.1 (optional)
V1.26.0	• docker.io/netapp/Trident:23.01.1 • docker.io/netapp/Trident-AutoSupport:23.01 • Registry.k8s.io/SIG-Storage/csi-provisioner:v3.4.0 • Registry.k8s.io/SIG-Storage/csi-Attacher:v4.1.0 • Registry.k8s.io/SIG-Storage/csi-resizer:v1.7.0 • Registry.k8s.io/SIG-Storage/csi-snapshotter:v6.2.1 • Registry.k8s.io/SIG-Storage/csi-Node-driver-Registrar:v2.7.0 • docker.io/netapp/Trident-Operator:23.01.1 (optional)

Verwenden Sie in Kubernetes ab Version 1.21 das validierte `registry.k8s.gcr.io/sig-storage/csi-snapshotter:v6.x` Bild nur, wenn der `v1` Version stellt den bereit `volumesnapshots.snapshot.storage.k8s.gcr.io` CRD.- Wenn der `v1beta1` Die Version dient der CRD mit/ohne dem `v1` Verwenden Sie die validierte Version `registry.k8s.gcr.io/sig-storage/csi-snapshotter:v3.x` Bild:

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 installieren `tridentctl`. In diesem Thema finden Sie wichtige Informationen zur Auswahl des richtigen Installationsprozesses.

Entscheidende Informationen zu Astra Trident 23.01

Sie müssen die folgenden wichtigen Informationen über Astra Trident lesen.

** Informationen über Astra TripereIT **

- Kubernetes 1.26 wird jetzt in Trident unterstützt. Upgrade von Trident vor dem Upgrade von Kubernetes.
- Astra Trident setzt die Verwendung von Multipathing-Konfiguration in SAN-Umgebungen strikt um und empfiehlt den Nutzen von `find_multipaths: no` In Multipath.conf Datei.

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

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

** Verheilen cappecackeIT **

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 Knotendemonsatz), 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.

** 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.

Methode	Installationsmodus
Trident-Operator (manuell)	"Standardinstallation" "Offline-Installation"

Methode	Installationsmodus
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".

Entscheidende Informationen zu Astra Trident 23.01

Sie müssen die folgenden wichtigen Informationen über Astra Trident lesen.

** Informationen über Astra TripereIT **

- Kubernetes 1.26 wird jetzt in Trident unterstützt. Upgrade von Trident vor dem Upgrade von Kubernetes.
- Astra Trident setzt die Verwendung von Multipathing-Konfiguration in SAN-Umgebungen strikt um und empfiehlt den Nutzen von `find_multipaths: no` In Multipath.conf Datei.

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

Trident-Operator kann manuell implementiert und Trident installiert werden

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

Bevor Sie beginnen

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

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

1. Überprüfen Sie Ihre Kubernetes Version:

```
kubectl version
```

2. Überprüfung der Berechtigungen für Cluster-Administratoren:

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

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

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

Schritt 1: Laden Sie das Trident Installer-Paket herunter

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

```
wget https://github.com/NetApp/trident/releases/download/v23.01.1/trident-
installer-23.01.1.tar.gz
tar -xf trident-installer-23.01.1.tar.gz
cd trident-installer
```

Schritt 2: Erstellen Sie die `TridentOrchestrator` CRD.

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

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

Schritt 3: Implementieren Sie den Trident-Operator

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

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

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

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

Schritte

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

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

> Um den Operator in einem anderen Namespace als dem bereitzustellen `trident` Namespace, Update `serviceaccount.yaml`, `clusterrolebinding.yaml` Und `operator.yaml` Und erstellen Sie Ihre Bundle-Datei mit `kustomization.yaml`:
>
> ```
> kubectl kustomize deploy/ > deploy/<bundle>.yaml
> ```

2. Überprüfen Sie, ob der Bediener bereitgestellt wurde.

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

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

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

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

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

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

kubectl describe torc trident

Name:        trident
Namespace:
Labels:      <none>
Annotations: <none>
API Version: trident.netapp.io/v1
Kind:        TridentOrchestrator
...
Spec:
  Debug:     true
  Namespace: trident
Status:
  Current Installation Params:
    IPv6:                      false
    Autosupport Hostname:
    Autosupport Image:         netapp/trident-autosupport:23.01
    Autosupport Proxy:
    Autosupport Serial Number:
    Debug:                     true
    Image Pull Secrets:
    Image Registry:
    k8sTimeout:           30
    Kubelet Dir:          /var/lib/kubelet
    Log Format:           text
    Silence Autosupport:  false
    Trident Image:        netapp/trident:23.01.1
  Message:                  Trident installed  Namespace:
trident
  Status:                   Installed
  Version:                  v23.01.1
Events:
    Type Reason Age From Message ---- ------ ---- ---- -------Normal
    Installing 74s trident-operator.netapp.io Installing Trident Normal
    Installed 67s trident-operator.netapp.io Trident installed
```

Überprüfen Sie die Installation

Die Installation kann auf verschiedene Weise überprüft werden.

Wird Verwendet `TridentOrchestrator` Status

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

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

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

Den Status der Pod-Erstellung verwenden

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

```
kubectl get pods -n trident

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

Wird Verwendet `tridentctl`

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

```
./tridentctl -n trident version

+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 23.01.1        | 23.01.1        |
+----------------+----------------+
```

Wie es weiter geht

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

Manuelles Bereitstellen des Trident-Mitarbeiters (Offline-Modus)

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

Entscheidende Informationen zu Astra Trident 23.01

Sie müssen die folgenden wichtigen Informationen über Astra Trident lesen.

Informationen über Astra TripereIT

- Kubernetes 1.26 wird jetzt in Trident unterstützt. Upgrade von Trident vor dem Upgrade von Kubernetes.
- Astra Trident setzt die Verwendung von Multipathing-Konfiguration in SAN-Umgebungen strikt um und empfiehlt den Nutzen von `find_multipaths: no` In Multipath.conf Datei.

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

Trident-Operator kann manuell implementiert und Trident installiert werden

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

Bevor Sie beginnen

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

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

1. Überprüfen Sie Ihre Kubernetes Version:

```
kubectl version
```

2. Überprüfung der Berechtigungen für Cluster-Administratoren:

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

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

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

Schritt 1: Laden Sie das Trident Installer-Paket herunter

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

```
wget https://github.com/NetApp/trident/releases/download/v23.01.1/trident-
installer-23.01.1.tar.gz
tar -xf trident-installer-23.01.1.tar.gz
cd trident-installer
```

Schritt 2: Erstellen Sie die `TridentOrchestrator` CRD.

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

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

Schritt 3: Aktualisieren Sie den Registrierungsort im Operator

In `/deploy/operator.yaml`, Update `image: docker.io/netapp/trident-operator:23.01.1` So geben Sie den Speicherort Ihrer Bildregistrierung an. Ihr "Trident und CSI-Images" Kann in einer Registrierung

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

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

Schritt 4: Implementieren des Trident-Operators

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

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

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

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

> Das Astra Trident-Installationsprogramm stellt eine Paketdatei bereit, mit der der Operator installiert und zugehörige Objekte erstellt werden können. Die Bundle-Datei ist eine einfache Möglichkeit, den Operator zu implementieren und Astra Trident mit einer Standardkonfiguration zu installieren.
>
> - Verwenden Sie für Cluster mit Kubernetes 1.24 oder älter `bundle_pre_1_25.yaml`.
> - Verwenden Sie für Cluster mit Kubernetes 1.25 oder höher `bundle_post_1_25.yaml`.

2. Überprüfen Sie, ob der Bediener bereitgestellt wurde.

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

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

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

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

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

Bilder in einer Registrierung

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

Bilder in verschiedenen Registern

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

```
imageRegistry: "<your-registry>/sig-storage"
autosupportImage: "<your-registry>/netapp/trident-autosupport:23.01"
tridentImage: "<your-registry>/netapp/trident:23.01.1"
```

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

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

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

kubectl describe torc trident

Name:        trident
Namespace:
Labels:      <none>
Annotations: <none>
API Version: trident.netapp.io/v1
Kind:        TridentOrchestrator
...
Spec:
  Autosupport Image:  <your-registry>/netapp/trident-autosupport:23.01
  Debug:              true
  Image Registry:     <your-registry>/sig-storage
  Namespace:          trident
  Trident Image:      <your-registry>/netapp/trident:23.01.1
Status:
  Current Installation Params:
    IPv6:                       false
    Autosupport Hostname:
    Autosupport Image:          <your-registry>/netapp/trident-
autosupport:23.01
    Autosupport Proxy:
    Autosupport Serial Number:
    Debug:                      true
    Http Request Timeout:       90s
    Image Pull Secrets:
    Image Registry:       <your-registry>/sig-storage
    k8sTimeout:           30
    Kubelet Dir:          /var/lib/kubelet
    Log Format:           text
    Probe Port:           17546
    Silence Autosupport:  false
    Trident Image:        <your-registry>/netapp/trident:23.01.1
  Message:                Trident installed
  Namespace:              trident
  Status:                 Installed
  Version:                v23.01.1
Events:
    Type Reason Age From Message ---- ------ ---- ---- -------Normal
    Installing 74s trident-operator.netapp.io Installing Trident Normal
    Installed 67s trident-operator.netapp.io Trident installed
```

Überprüfen Sie die Installation

Die Installation kann auf verschiedene Weise überprüft werden.

Wird Verwendet `TridentOrchestrator` Status

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

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

Den Status der Pod-Erstellung verwenden

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

```
kubectl get pods -n trident

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

Wird Verwendet `tridentctl`

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

```
./tridentctl -n trident version

+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 23.01.1        | 23.01.1        |
+----------------+----------------+
```

Wie es weiter geht

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

Trident Operator mit Helm (Standard-Modus) implementieren

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

Entscheidende Informationen zu Astra Trident 23.01

Sie müssen die folgenden wichtigen Informationen über Astra Trident lesen.

** Informationen über Astra TripereIT **

- Kubernetes 1.26 wird jetzt in Trident unterstützt. Upgrade von Trident vor dem Upgrade von Kubernetes.
- Astra Trident setzt die Verwendung von Multipathing-Konfiguration in SAN-Umgebungen strikt um und empfiehlt den Nutzen von `find_multipaths: no` In Multipath.conf Datei.

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

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

Verwendung von Trident "Steuerruderdiagramm" Sie können den Trident Operator implementieren und Trident in einem Schritt installieren.

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

Bevor Sie beginnen

Zusätzlich zum "Voraussetzungen für die Implementierung" Die Sie benötigen "Helm Version 3".

Schritte

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

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

2. Nutzung `helm install` Und geben Sie einen Namen für Ihre Bereitstellung an, wie im folgenden Beispiel, wo `23.01.1` Ist die Version des Astra Trident, die Sie installieren.

```
helm install <name> netapp-trident/trident-operator --version 23.01.1
--create-namespace --namespace <trident-namespace>
```

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

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

Konfigurationsdaten während der Installation übergeben

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

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

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

```
helm install <name> netapp-trident/trident-operator --version 23.01.1
--create-namespace --namespace --set tridentDebug=true
```

Konfigurationsoptionen

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

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

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

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

Allgemeines zu Controller-Pods und Node-Pods

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

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

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

Wie es weiter geht

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

Trident-Operator mit Helm (Offline-Modus) implementieren

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

Entscheidende Informationen zu Astra Trident 23.01

Sie müssen die folgenden wichtigen Informationen über Astra Trident lesen.

** Informationen über Astra TriperelT **

- Kubernetes 1.26 wird jetzt in Trident unterstützt. Upgrade von Trident vor dem Upgrade von Kubernetes.
- Astra Trident setzt die Verwendung von Multipathing-Konfiguration in SAN-Umgebungen strikt um und empfiehlt den Nutzen von `find_multipaths: no` In Multipath.conf Datei.

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

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

Verwendung von Trident "Steuerruderdiagramm" Sie können den Trident Operator implementieren und Trident in einem Schritt installieren.

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

Bevor Sie beginnen

Zusätzlich zum "Voraussetzungen für die Implementierung" Die Sie benötigen "Helm Version 3".

Schritte

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

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

2. Nutzung `helm install` Und geben Sie einen Namen für die Bereitstellung und den Speicherort der Image-Registrierung an. Ihr "Trident und CSI-Images" Kann in einer Registrierung oder in verschiedenen Registern gefunden werden, aber alle CSI-Images müssen sich in derselben Registrierung befinden. In den Beispielen: `23.01.1` Ist die Version des Astra Trident, die Sie installieren.

Bilder in einer Registrierung

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

Bilder in verschiedenen Registern

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

```
helm install <name> netapp-trident/trident-operator --version
23.01.1 --set imageRegistry=<your-registry>/sig-storage --set
operatorImage=<your-registry>/netapp/trident-operator:23.01.1 --set
tridentAutosupportImage=<your-registry>/netapp/trident-
autosupport:23.01 --set tridentImage=<your-
registry>/netapp/trident:23.01.1 --create-namespace --namespace
<trident-namespace>
```

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

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

Konfigurationsdaten während der Installation übergeben

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

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

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

```
helm install <name> netapp-trident/trident-operator --version 23.01.1
--create-namespace --namespace --set tridentDebug=true
```

Konfigurationsoptionen

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

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

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

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

Allgemeines zu Controller-Pods und Node-Pods

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

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

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

Wie es weiter geht

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

Anpassen der Trident Operator-Installation

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

Allgemeines zu Controller-Pods und Node-Pods

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

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

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

Konfigurationsoptionen

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

Diese Tabelle enthält Einzelheiten `TridentOrchestrator` Attribute.

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

Parameter	Beschreibung	Standard
`imagePullPolicy`	Legt die BildPull-Richtlinie für den Trident-Operator fest. Gültige Werte sind: `Always` Um immer das Bild zu ziehen. `IfNotPresent` Nur wenn das Image nicht auf dem Node vorhanden ist, soll das Image kopiert werden. `Never` Nie das Bild ziehen.	`IfNotPresent`
`controllerPluginNodeSelector`	Zusätzliche Node-Auswahl für Pods Entspricht dem gleichen Format wie pod.spec.nodeSelector.	Kein Standard; optional
`controllerPluginTolerations`	Überschreibt Kubernetes-Toleranzen für Pods. Entspricht dem gleichen Format wie pod.spec.tolerations.	Kein Standard; optional
`nodePluginNodeSelector`	Zusätzliche Node-Auswahl für Pods Entspricht dem gleichen Format wie pod.spec.nodeSelector.	Kein Standard; optional
`nodePluginTolerations`	Überschreibt Kubernetes-Toleranzen für Pods. Entspricht dem gleichen Format wie pod.spec.tolerations.	Kein Standard; optional

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

Beispielkonfigurationen

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

Beispiel 1: Grundlegende benutzerdefinierte Konfiguration

Dies ist ein Beispiel für eine benutzerdefinierte Grundkonfiguration.

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

Beispiel 2: Implementierung mit Node-Auswahl

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

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

Beispiel 3: Bereitstellung auf Windows Worker-Nodes

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

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

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".

Entscheidende Informationen zu Astra Trident 23.01

Sie müssen die folgenden wichtigen Informationen über Astra Trident lesen.

** Informationen über Astra TriperelT **

- Kubernetes 1.26 wird jetzt in Trident unterstützt. Upgrade von Trident vor dem Upgrade von Kubernetes.
- Astra Trident setzt die Verwendung von Multipathing-Konfiguration in SAN-Umgebungen strikt um und empfiehlt den Nutzen von `find_multipaths: no` In Multipath.conf Datei.

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

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/v23.01.1/trident-
installer-23.01.1.tar.gz
tar -xf trident-installer-23.01.1.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.

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

Standardmodus

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

Bilder in einer Registrierung

```
./tridentctl install -n trident --image-registry <your-registry>
--autosupport-image <your-registry>/trident-autosupport:23.01 --trident
-image <your-registry>/trident:23.01.1
```

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

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=23.01.1
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 |
+----------------+----------------+
| 23.01.1        | 23.01.1        |
+----------------+----------------+
```

Wie es weiter geht

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

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

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

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

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

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

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

Was kommt als Nächstes?

Nach der Installation von Astra Trident können Sie mit dem Erstellen eines Backends fortfahren, eine Storage-Klasse erstellen, ein Volume bereitstellen und das Volume in einem Pod mounten.

Schritt 1: Erstellen Sie ein Backend

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

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

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

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

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

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

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

Schritt 2: Erstellen Sie eine Storage-Klasse

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

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

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

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

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

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

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

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

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

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

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

Schritt 3: Stellen Sie Ihr erstes Volumen bereit

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

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

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

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

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

Schritt 4: Mounten Sie die Volumes in einem POD

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

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

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

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

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

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

Löschen Sie zum Löschen des Volumes die Forderung:

```
kubectl delete pvc basic
```

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

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

Managen Sie 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 sowie Bug Fixes und Verbesserungen. Wir empfehlen Ihnen, ein Upgrade mindestens einmal pro Jahr durchzuführen, um von den neuen Funktionen in Astra Trident zu profitieren.

Wählen Sie eine Version aus

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 22.01 auf 23.01 aktualisieren (einschließlich aller Punktversionen, z. B. 22.01.1).
- Wenn Sie eine frühere Version haben, sollten Sie eine mehrstufige Aktualisierung anhand der Dokumentation der jeweiligen Version durchführen, um spezifische Anweisungen zu erhalten. Dazu müssen Sie zuerst ein Upgrade auf die aktuellste Version durchführen, die zu Ihrem vier-Release-Fenster passt. Wenn Sie z. B. 18.07 ausführen und auf Version 20.07 aktualisieren möchten, befolgen Sie den Prozess zum mehrstufigen Upgrade wie unten angegeben:
  a. Erstes Upgrade von 18.07 auf 19.07.
  b. Dann aktualisieren Sie von 19.07 auf 20.07.

> - Für alle Upgrades von Version 19.04 und früher ist die Migration der Astra Trident Metadaten von alleine erforderlich `etcd` Zu CRD-Objekten. Prüfen Sie die Dokumentation der Version, um zu verstehen, wie das Upgrade funktioniert.
> - 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 von security Kontexten 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`^]. Weitere Informationen finden Sie unter "Bekannte Probleme".

Wählen Sie eine Upgrade-Option aus

Astra Trident bietet zwei Optionen für ein Upgrade. Im Allgemeinen verwenden Sie die gleiche Option, die Sie für die Erstinstallation verwendet haben, wie Sie jedoch können "Wechseln Sie zwischen den Installationsmethoden".

- "Upgrade über den Trident-Operator"
*

> CSI Volume Snapshots ist jetzt eine GA-Funktion, die ab Kubernetes 1.20 beginnt. Beim Upgrade von Astra Trident müssen alle vorherigen Alpha-Snapshot-CRS und CRDs (Volume Snapshot-Klassen, Volume-Snapshots und Volume Snapshot-Inhalt) entfernt werden, bevor das Upgrade durchgeführt wird. Siehe "Diesem Blog" Schritte zur Migration von Alpha-Snapshots in die Beta/GA-Spezifikation

Änderungen am Operator

In der Version 21.01 von Astra Trident werden dem Betreiber einige wichtige architektonische Änderungen vorgestellt:

- Der Operator ist jetzt **Cluster-scoped**. Vorherige Instanzen des Trident Operators (Versionen 20.04 bis 20.10) waren **Namespace-Scoped**. Ein Operator mit Cluster-Scoped ist aus den folgenden Gründen von Vorteil:
  - Resource Accountability: Der Operator managt jetzt die mit einer Astra Trident-Installation verbundenen Ressourcen auf Cluster-Ebene. Im Rahmen der Installation von Astra Trident erstellt und verwaltet der Bediener mehrere Ressourcen mit `ownerReferences`. Wartung `ownerReferences` Auf Cluster-Scoped-Ressourcen können Fehler bei bestimmten Kubernetes-Distributoren wie OpenShift auftreten. Diese Option wird durch einen Operator mit Cluster-Umfang entschärft. Für die automatische Reparatur und das Patching von Trident-Ressourcen ist dies eine wesentliche Anforderung.
  - Aufräumarbeiten während der Deinstallation: Eine vollständige Entfernung von Astra Trident würde alle damit verbundenen Ressourcen zu löschen benötigen. Ein Operator mit Namespace-Scoped kann Probleme beim Entfernen von Cluster-Scoped-Ressourcen (wie clusterRole, ClusterRoleBending und PodSecurityPolicy) haben und eine unvollständige Bereinigung zur Folge haben. Ein Operator mit Cluster-Umfang beseitigt dieses Problem. Benutzer können Astra Trident vollständig deinstallieren und bei Bedarf neu installieren.
- `TridentProvisioner` Wird nun durch ersetzt `TridentOrchestrator` Als benutzerdefinierte Ressource, die für die Installation und das Management von Astra Trident verwendet wird. Darüber hinaus wird dem ein neues Feld vorgestellt `TridentOrchestrator` Spez. Benutzer können angeben, dass der Namespace Trident über den installiert/aktualisiert werden muss `spec.namespace` Feld. Sie können sich ein Beispiel ansehen "Hier".

Upgrade mit dem Bediener

Sie können eine bestehende Astra Trident-Installation ganz einfach mithilfe des Betreibers aufrüsten.

Bevor Sie beginnen

Um mit dem Bediener zu aktualisieren, sollten die folgenden Bedingungen erfüllt sein:

- Sie benötigen eine CSI-basierte Astra Trident-Installation. Alle Versionen von 19.07 sind CSI-basiert. Sie können die Pods in Ihrem Trident Namespace überprüfen und überprüfen.
  - Die Benennung von Pods in Versionen vor 23.01 folgt A `trident-csi-*` übereinkommen.
  - Pod-Benennungen in 23.01 und höher: `trident-controller-<generated id>` Für den Controller POD `trident-node-<operating system>-<generated id>` Für die Node-Pods: `trident-operator-<generated id>` Für den Bediener-POD.

- Wenn Sie CSI Trident deinstalliert haben und die Metadaten aus der Installation beibehalten werden, können Sie mithilfe des Operators ein Upgrade durchführen.
- Es sollte nur eine Astra Trident Installation über alle Namespaces in einem bestimmten Kubernetes Cluster hinweg vorhanden sein.
- Ein Kubernetes-Cluster sollte ausgeführt werden "Eine unterstützte Kubernetes Version".
- Wenn Alpha-Snapshot-CRDs vorhanden sind, sollten Sie sie mit entfernen `tridentctl obliviate alpha-snapshot-crd`. Dadurch werden die CRDs für die Alpha-Snapshot-Spezifikation gelöscht. Informationen zu vorhandenen Snapshots, die gelöscht/migriert werden sollen, finden Sie unter "Diesem Blog".

- Wenn Sie das Upgrade von Trident mithilfe des Betreibers für OpenShift Container Platform vornehmen, sollten Sie ein Upgrade auf Trident 21.01.1 oder höher durchführen. 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".
- Verwenden Sie den Operator zum Aktualisieren von Trident nicht, wenn Sie ein verwenden `etcd`-Based Trident Release (19.04 oder früher).

Upgrade einer Installation des Trident Operators mit Cluster-Umfang

Führen Sie die folgenden Schritte aus, um eine Installation des Trident Operators mit Cluster-Umfang zu aktualisieren. Alle Astra Trident Versionen 21.01 und höher verwenden einen Operator mit Cluster-Umfang.

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 22.01 durchführen, führen Sie den folgenden Befehl aus:

```
kubectl delete -f 22.01/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 Paket-YAML-Datei für Ihre Umgebung und Astra Trident-Version. Wenn Sie beispielsweise Astra Trident 23.01 für Kubernetes 1.26 installieren, führen Sie den folgenden Befehl aus:

```
kubectl create -f 23.01.1/trident-installer/deploy/bundle_post_1_25.yaml
-n trident
```

Trident stellt eine Paketdatei bereit, mit der der Operator installiert und damit verknüpfte Objekte für die Kubernetes-Version erstellt werden können.

- Verwenden Sie für Cluster mit Kubernetes 1.24 oder älter "Bundle_pre_1_25.yaml".
- Verwenden Sie für Cluster mit Kubernetes 1.25 oder höher "Bundle_Post_1_25.yaml".

Ergebnisse

Der Trident-Operator identifiziert eine bestehende Astra Trident-Installation und aktualisiert sie auf die gleiche Version wie der Operator.

Aktualisieren einer Installation des Namespace-Scoped-Operators

Führen Sie die folgenden Schritte aus, um von einer Instanz von Astra Trident zu aktualisieren, die mit dem Namespace-Scoped Operator (Versionen 20.07 bis 20.10) installiert wurde.

Schritte

1. Überprüfen Sie den Status der vorhandenen Trident Installation. Prüfen Sie dazu den **Status** von `TridentProvisioner`. Der Status sollte sein `Installed`.

```
kubectl describe tprov trident -n trident | grep Message: -A 3
Message:  Trident installed
Status:   Installed
Version:  v20.10.1
```

Wenn der Status angezeigt wird `Updating`, Stellen Sie sicher, dass Sie es lösen, bevor Sie fortfahren. Eine Liste möglicher Statuswerte finden Sie unter "Hier".

2. Erstellen Sie die `TridentOrchestrator` CRD mit dem Manifest, das mit dem Trident-Installer bereitgestellt wurde.

```
# Download the release required [23.01.1]
mkdir 23.01.1
cd 23.01.1
wget
https://github.com/NetApp/trident/releases/download/v23.01.1/trident-
installer-23.01.1.tar.gz
tar -xf trident-installer-23.01.1.tar.gz
cd trident-installer
kubectl create -f
deploy/crds/trident.netapp.io_tridentorchestrators_crd_post1.16.yaml
```

3. Löschen Sie den Operator Namespace-Scoped mithilfe des Manifests. Um diesen Schritt abzuschließen, benötigen Sie die Paket-YAML-Datei, die zum Bereitstellen des Namespace-Scoped-Operators von verwendet wird `https://github.com/NetApp/trident/tree/stable/`*`vXX.XX`* `/deploy/`*`BUNDLE.YAML`* Wo *`vXX.XX`* Ist die Versionsnummer und *`BUNDLE.YAML`* Ist der Name der Paket-YAML-Datei.

Sie sollten die erforderlichen Änderungen an den Trident Installationsparametern vornehmen (z. B. Ändern der Werte für `tridentImage`, `autosupportImage`, Ein privates Image Repository von und das `imagePullSecrets`) Nach dem Löschen des Operator Namespace-scoped und vor der Installation des Operators Cluster-scoped. Eine vollständige Liste der Parameter, die aktualisiert werden können, finden Sie im "Konfigurationsoptionen".

```
#Ensure you are in the right directory
pwd
/root/20.10.1/trident-installer

#Delete the namespace-scoped operator
kubectl delete -f deploy/<BUNDLE.YAML> -n trident
serviceaccount "trident-operator" deleted
clusterrole.rbac.authorization.k8s.io "trident-operator" deleted
clusterrolebinding.rbac.authorization.k8s.io "trident-operator" deleted
deployment.apps "trident-operator" deleted
podsecuritypolicy.policy "tridentoperatorpods" deleted

#Confirm the Trident operator was removed
kubectl get all -n trident
NAME                                 READY   STATUS    RESTARTS   AGE
pod/trident-csi-68d979fb85-dsrmn     6/6     Running   12         99d
pod/trident-csi-8jfhf                2/2     Running   6          105d
pod/trident-csi-jtnjz                2/2     Running   6          105d
pod/trident-csi-lcxvh                2/2     Running   8          105d

NAME                  TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)
AGE
service/trident-csi   ClusterIP   10.108.174.125   <none>
34571/TCP,9220/TCP   105d

NAME                           DESIRED   CURRENT   READY   UP-TO-DATE
AVAILABLE   NODE SELECTOR                                        AGE
daemonset.apps/trident-csi   3         3         3       3            3
kubernetes.io/arch=amd64,kubernetes.io/os=linux   105d

NAME                           READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/trident-csi    1/1     1            1           105d

NAME                                      DESIRED   CURRENT   READY   AGE
replicaset.apps/trident-csi-68d979fb85    1         1         1
105d
```

In dieser Phase, der `trident-operator-xxxxxxxxxx-xxxxx` Pod wurde gelöscht.

4. (Optional) Wenn die Installationsparameter geändert werden müssen, aktualisieren Sie den `TridentProvisioner` Spez. Dies können Änderungen sein, wie z. B. das Ändern der privaten Image-Registry zum Ziehen von Container-Images, das Aktivieren von Debug-Protokollen oder das Festlegen von Image Pull Secrets.

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

5. Installieren Sie den Trident Operator.

Durch die Installation des Operators Cluster-Scoped wird die Migration von initiiert `TridentProvisioner` Objekte an `TridentOrchestrator` Objekte, löscht `TridentProvisioner` Objekte und das `tridentprovisioner` CRD, und aktualisiert Astra Trident auf die Version des verwendeten Cluster-Scoped-Betreibers. Im folgenden Beispiel wurde für Trident ein Upgrade auf 23.01.1 durchgeführt.

Ein Upgrade des Astra Trident mithilfe des Trident Operators führt zu einer Migration von `tridentProvisioner` Zu A `tridentOrchestrator` Objekt mit dem gleichen Namen. Dieser Vorgang wird automatisch vom Betreiber übernommen. Auch Astra Trident ist auf dem Upgrade im selben Namespace wie zuvor installiert.

```
#Ensure you are in the correct directory
pwd
/root/23.01.1/trident-installer

#Install the cluster-scoped operator in the **same namespace**
kubectl create -f deploy/<BUNDLE.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

#All tridentProvisioners will be removed, including the CRD itself
kubectl get tprov -n trident
Error from server (NotFound): Unable to list "trident.netapp.io/v1,
Resource=tridentprovisioners": the server could not find the requested
resource (get tridentprovisioners.trident.netapp.io)

#tridentProvisioners are replaced by tridentOrchestrator
kubectl get torc
NAME      AGE
trident   13s

#Examine Trident pods in the namespace
kubectl get pods -n trident
NAME                                READY   STATUS    RESTARTS
AGE
trident-controller-79df798bdc-m79dc      6/6     Running   0
1m41s
trident-node-linux-xrst8                 2/2     Running   0
1m41s
trident-operator-5574dbbc68-nthjv        1/1     Running   0
1m52s

#Confirm Trident has been updated to the desired version
kubectl describe torc trident | grep Message -A 3
Message:                Trident installed
Namespace:              trident
Status:                 Installed
Version:                v23.01.1
```

Der `trident-controller` Zudem geben die Pod-Namen die im Jahr 23.01 eingeführte Namenskonvention wieder.

Aktualisieren einer Helm-basierten Bedienerinstallation

Führen Sie die folgenden Schritte durch, um eine Helm-basierte Bedienerinstallation zu aktualisieren.

> 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-23.01.1.tgz` Gibt die Version an, auf die Sie ein Upgrade durchführen möchten.

```
helm upgrade <name> trident-operator-23.01.1.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 zu ändern `tridentDebug`, Ausführen des folgenden Befehls:
>
> ```
> helm upgrade <name> trident-operator-23.01.1-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.

Ergebnisse

Der Trident-Operator identifiziert eine bestehende Astra Trident-Installation und aktualisiert sie auf die gleiche Version wie der Operator.

Upgrade von einer nicht-Betreiber-Installation

Sie können ein Upgrade auf die neueste Version des Trident-Operators von durchführen `tridentctl` Installation:

Schritte

1. Laden Sie die neueste Version von Astra Trident herunter.

```
# Download the release required [23.01.1]
mkdir 23.01.1
cd 23.01.1
wget
https://github.com/NetApp/trident/releases/download/v22.01.1/trident-
installer-23.01.1.tar.gz
tar -xf trident-installer-23.01.1.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 Bediener bereit.

```
#Install the cluster-scoped operator in the **same namespace**
kubectl create -f deploy/<BUNDLE.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.

```
#Create a tridentOrchestrator to initiate a Trident install
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

#Confirm Trident was upgraded to the desired version
kubectl describe torc trident | grep Message -A 3
Message:                Trident installed
Namespace:              trident
Status:                 Installed
Version:                v23.01.1
```

Ergebnisse

Die vorhandenen Back-Ends und PVCs stehen automatisch zur Verfügung.

Upgrade mit tridentctl

Sie können mithilfe von ganz einfach eine bestehende Astra Trident Installation aufrüsten `tridentctl`.

Überlegungen vor dem Upgrade

Bei einem Upgrade auf die neueste Version von Astra Trident sollten Sie Folgendes berücksichtigen:

- Ab Trident 20.01 gilt nur als Beta-Version von "Volume Snapshots" Wird unterstützt. Kubernetes-Administratoren sollten darauf achten, dass sie die Alpha-Snapshot-Objekte sicher in Beta-Version sichern oder in sie konvertieren können, um die älteren Alpha-Snapshots zu behalten.
- Die Beta-Version von Volume Snapshots führt einen geänderten Satz von CRDs und einen Snapshot-Controller ein, die beide vor der Installation von Astra Trident eingerichtet werden sollten. "Diesem Blog" Erläutert die Schritte, die bei der Migration von Alpha-Volume-Snapshots in das Beta-Format erforderlich sind.
- 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.

Unterbrechen Sie beim Upgrade von Astra Trident nicht den Upgrade-Prozess. Vergewissern Sie sich, dass das Installationsprogramm ausgeführt wird.

Nächste Schritte nach dem Upgrade

Um die umfangreichen Funktionen neuerer Trident Versionen (wie On-Demand Volume Snapshots) nutzen zu können, können Sie die Volumes mit der aktualisieren `tridentctl upgrade` Befehl.

Wenn es Legacy-Volumes gibt, sollten Sie diese von einem NFS/iSCSI-Typ auf den CSI-Typ upgraden, um die umfassenden neuen Funktionen in Astra Trident zu nutzen. Ein von Trident bereitgestelltes Legacy-PV unterstützt die herkömmlichen Funktionen.

Beachten Sie bei der Entscheidung, Volumes auf den CSI-Typ zu aktualisieren:

- Möglicherweise müssen Sie nicht alle Volumes aktualisieren. Zuvor erstellte Volumes sind weiterhin zugänglich und funktionieren ordnungsgemäß.
- Ein PV kann als Teil einer Deployment/StatupfulSet installiert werden. Es ist nicht erforderlich, die Implementierung/StatesSet herunterzufahren.
- Sie können ein PV nicht an einen eigenständigen Pod anschließen, wenn Sie ein Upgrade durchführen. Sie sollten den POD herunterfahren, bevor Sie das Volume aktualisieren.
- Sie können nur ein Volume aktualisieren, das an eine PVC gebunden ist. Volumes, die nicht an PVCs gebunden sind, sollten vor dem Upgrade entfernt und importiert werden.

Beispiel für ein Volume-Upgrade

Dies ist ein Beispiel, das die Durchführung eines Volume-Upgrades zeigt.

1. Laufen `kubectl get pv` So Listen Sie die PVS auf.

```
kubectl get pv
NAME                        CAPACITY     ACCESS MODES   RECLAIM POLICY
STATUS   CLAIM                   STORAGECLASS    REASON   AGE
default-pvc-1-a8475         1073741824   RWO            Delete
Bound    default/pvc-1          standard                19h
default-pvc-2-a8486         1073741824   RWO            Delete
Bound    default/pvc-2          standard                19h
default-pvc-3-a849e         1073741824   RWO            Delete
Bound    default/pvc-3          standard                19h
default-pvc-4-a84de         1073741824   RWO            Delete
Bound    default/pvc-4          standard                19h
trident                     2Gi          RWO            Retain
Bound    trident/trident                                19h
```

Derzeit gibt es vier PVS, die von Trident 20.07 mit dem erstellt wurden `netapp.io/trident` bereitstellung:

2. Laufen `kubectl describe pv` Um die Details zum PV zu erhalten.

```
kubectl describe pv default-pvc-2-a8486

Name:            default-pvc-2-a8486
Labels:          <none>
Annotations:     pv.kubernetes.io/provisioned-by: netapp.io/trident
                 volume.beta.kubernetes.io/storage-class: standard
Finalizers:      [kubernetes.io/pv-protection]
StorageClass:    standard
Status:          Bound
Claim:           default/pvc-2
Reclaim Policy:  Delete
Access Modes:    RWO
VolumeMode:      Filesystem
Capacity:        1073741824
Node Affinity:   <none>
Message:
Source:
    Type:      NFS (an NFS mount that lasts the lifetime of a pod)
    Server:    10.xx.xx.xx
    Path:      /trid_1907_alpha_default_pvc_2_a8486
    ReadOnly:  false
```

Das PV wurde mit Hilfe des erstellt `netapp.io/trident` bereitstellung vom Typ NFS. Um alle neuen Funktionen von Astra Trident zu unterstützen, sollte dieses PV auf den CSI-Typ aufgerüstet werden.

3. Führen Sie die aus `tridentctl upgrade volume <name-of-trident-volume>` Befehl zum Upgrade eines alten Astra Trident Volumes auf die CSI-Spezifikation.

```
./tridentctl get volumes -n trident
+---------------------+---------+---------------+----------
+--------------------------------------+--------+---------+
|            NAME     |  SIZE   | STORAGE CLASS | PROTOCOL |
BACKEND UUID             | STATE  | MANAGED |
+---------------------+---------+---------------+----------
+--------------------------------------+--------+---------+
| default-pvc-2-a8486 | 1.0 GiB | standard      | file     | c5a6f6a4-
b052-423b-80d4-8fb491a14a22 | online | true    |
| default-pvc-3-a849e | 1.0 GiB | standard      | file     | c5a6f6a4-
b052-423b-80d4-8fb491a14a22 | online | true    |
| default-pvc-1-a8475 | 1.0 GiB | standard      | file     | c5a6f6a4-
b052-423b-80d4-8fb491a14a22 | online | true    |
| default-pvc-4-a84de | 1.0 GiB | standard      | file     | c5a6f6a4-
b052-423b-80d4-8fb491a14a22 | online | true    |
+---------------------+---------+---------------+----------
+--------------------------------------+--------+---------+

./tridentctl upgrade volume default-pvc-2-a8486 -n trident
+---------------------+---------+---------------+----------
+--------------------------------------+--------+---------+
|            NAME     |  SIZE   | STORAGE CLASS | PROTOCOL |
BACKEND UUID             | STATE  | MANAGED |
+---------------------+---------+---------------+----------
+--------------------------------------+--------+---------+
| default-pvc-2-a8486 | 1.0 GiB | standard      | file     | c5a6f6a4-
b052-423b-80d4-8fb491a14a22 | online | true    |
+---------------------+---------+---------------+----------
+--------------------------------------+--------+---------+
```

4. A ausführen `kubectl describe pv` Um zu überprüfen, ob es sich bei dem Volumen um ein CSI-Volumen handelt.

```
kubectl describe pv default-pvc-2-a8486
Name:            default-pvc-2-a8486
Labels:          <none>
Annotations:     pv.kubernetes.io/provisioned-by: csi.trident.netapp.io
                 volume.beta.kubernetes.io/storage-class: standard
Finalizers:      [kubernetes.io/pv-protection]
StorageClass:    standard
Status:          Bound
Claim:           default/pvc-2
Reclaim Policy:  Delete
Access Modes:    RWO
VolumeMode:      Filesystem
Capacity:        1073741824
Node Affinity:   <none>
Message:
Source:
    Type:              CSI (a Container Storage Interface (CSI) volume
source)
    Driver:            csi.trident.netapp.io
    VolumeHandle:      default-pvc-2-a8486
    ReadOnly:          false
    VolumeAttributes:      backendUUID=c5a6f6a4-b052-423b-80d4-
8fb491a14a22

internalName=trid_1907_alpha_default_pvc_2_a8486
                           name=default-pvc-2-a8486
                           protocol=file
Events:                <none>
```

Auf diese Weise können Sie Volumes des von Astra Trident erstellten NFS-/iSCSI-Typs auf Basis der einzelnen Volumes auf CSI-Typ aufrüsten.

Deinstallieren Sie Astra Trident

Je nachdem, wie Astra Trident installiert ist, gibt es mehrere Optionen, um es zu deinstallieren.

Deinstallieren Sie mit Helm

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 die Deinstallation mit dem Trident-Operator

Wenn Sie Astra Trident über den Operator installiert haben, können Sie es deinstallieren, indem Sie einen der folgenden Schritte durchführen:

- **Bearbeiten `TridentOrchestrator` So legen Sie die Deinstallationsflag fest:** können Sie bearbeiten `TridentOrchestrator` Und gesetzt `spec.uninstall=true`. Bearbeiten Sie das `TridentOrchestrator` CR und stellen Sie den ein `uninstall` Flag wie unten gezeigt:

  ```
  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 des `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. Um den Astra Trident komplett zu entfernen (einschließlich der von ihm erstellten CRDs) und effektiv den Schiefer sauber zu löschen, können Sie bearbeiten `TridentOrchestrator` Um die zu bestehen `wipeout` Option. Das folgende Beispiel zeigt:

```
kubectl patch torc <trident-orchestrator-name> --type=merge -p
'{"spec":{"wipeout":["crds"],"uninstall":true}}'
```

Damit wird Astra Trident vollständig deinstalliert und alle Metadaten gelöscht, die mit den gemanagten Back-Ends und Volumes zusammenhängen. Nachfolgende Installationen werden als frische Installationen behandelt.

Sie sollten nur erwägen, die CRDs zu löschen, wenn Sie eine vollständige Deinstallation durchführen. Dieser Vorgang kann nicht rückgängig gemacht werden. **Wischen Sie die CRDs nicht ab, es sei denn, Sie möchten von vorne beginnen und eine neue Astra Trident Installation erstellen**.

Deinstallieren Sie mit `tridentctl`

Führen Sie die aus `uninstall` Befehl in `tridentctl` Wie folgt entfernt alle mit Astra Trident verbundenen Ressourcen außer den CRDs und verwandten Objekten, so dass es einfach ist, das Installationsprogramm erneut auszuführen, um auf eine neuere Version zu aktualisieren.

```
./tridentctl uninstall -n <namespace>
```

Um Astra Trident vollständig zu entfernen, sollten Sie die Finalizer für die von Astra Trident erstellten CRDs entfernen und die CRDs löschen.

Downgrade Astra Trident

Erfahren Sie mehr über die Schritte beim Downgrade auf eine frühere Version von Astra Trident.

Wenn sie heruntergestuft werden müssen

Sie können aus verschiedenen Gründen ein Downgrade in Betracht ziehen, beispielsweise aus folgenden Gründen:

- Verfügbarkeitsplanung
- Sofortige Behebung von Fehlern, die als Folge eines Upgrades beobachtet wurden
- Abhängigkeitsprobleme, nicht erfolgreiche und unvollständige Upgrades

Beim Wechsel zu einer Astra Trident-Version mit CRDs sollten Sie ein Downgrade in Betracht ziehen. Astra Trident verwendet CRDs zur Statuswahrung. Daher verfügen alle erstellten Storage-Einheiten (Back-Ends, Storage-Klassen, PV und Volume Snapshots) über zugehörige CRD-Objekte anstatt in die geschriebene Daten `trident` PV (verwendet von der früheren installierten Version von Astra Trident). Neu erstellte PVS, Back-Ends und Storage-Klassen werden als CRD-Objekte verwaltet.

Nur eine Downgrade für eine Version von Astra Trident versuchen, die mit CRDs (19.07 und höher) ausgeführt wird. Dadurch wird sichergestellt, dass die im aktuellen Astra Trident Release ausgeführten Vorgänge nach dem Downgrade sichtbar sind.

Wenn Sie nicht herunterstufen

Sie sollten kein Downgrade auf eine Version von Trident durchführen, die verwendet wird `etcd` Zustand beibehalten (19.04 und früher). Alle Operationen, die mit der aktuellen Astra Trident Version ausgeführt werden, werden nach der Herabstufung nicht berücksichtigt. Neu erstellte PVS können nicht verwendet werden, wenn Sie zurück auf eine frühere Version wechseln. Änderungen an Objekten wie Back-Ends, PVS, Storage-Klassen und Volume Snapshots (erstellt/aktualisiert/gelöscht) sind für Astra Trident nicht sichtbar, wenn sie zurück auf eine frühere Version verschoben werden. Wenn Sie zu einer früheren Version zurückkehren, wird der Zugriff auf PVS, die bereits mit der älteren Version erstellt wurden, nicht unterbrochen,

es sei denn, sie wurden aktualisiert.

Downgrade bei der Installation von Astra Trident mit dem Operator

Bei Installationen, die mit dem Trident Operator abgeschlossen wurden, ist das Downgrade-Verfahren anders und erfordert nicht die Verwendung von `tridentctl`.

Bei Installationen, die mit dem Trident-Operator durchgeführt wurden, kann Astra Trident auf eine der folgenden Werte heruntergestuft werden:

- Eine Version, die mithilfe des Namespace-Scoped-Operators installiert wird (20.07 - 20.10).
- Eine Version, die mit dem Cluster-Scoped Operator (21.01 und höher) installiert wird.

Downgrade auf einen Operator mit Cluster-Umfang

Führen Sie zum Downgrade von Astra Trident auf eine Version mit dem Operator Cluster-Scoped die unten aufgeführten Schritte aus.

Schritte

1. "Deinstallieren Sie Astra Trident". **Löschen Sie die CRDs nicht, es sei denn, Sie möchten eine vorhandene Installation vollständig entfernen.**
2. Der Trident-Operator kann mithilfe des Operator-Manifests gelöscht werden, das mit Ihrer Trident-Version verknüpft ist. Beispiel: `https://github.com/NetApp/trident/tree/stable/vXX.XX /deploy/bundle.yaml` Wo *`vXX.XX`* Ist die Versionsnummer (zum Beispiel `v22.10`) Und *`bundle.yaml`* Ist der Name der Paket-YAML-Datei.
3. Fahren Sie mit der Installation der gewünschten Version von Astra Trident fort. Befolgen Sie die Dokumentation für das gewünschte Release.

Downgrade auf Operator mit Namespace-Scoped

In diesem Abschnitt werden die Schritte für die Herabstufung auf eine Astra Trident-Version zusammengefasst, die im Bereich 20.07 bis 20.10 liegt, die über den Operator Namespace-Scoped installiert werden.

Schritte

1. "Deinstallieren Sie Astra Trident". **Die CRDs dürfen nicht gelöscht werden, es sei denn, Sie möchten eine vorhandene Installation vollständig entfernen.** stellen Sie sicher, dass die `tridentorchestrator` Wird gelöscht.

```
#Check to see if there are any tridentorchestrators present
kubectl get torc
NAME        AGE
trident     20h

#Looks like there is a tridentorchestrator that needs deleting
kubectl delete torc trident
tridentorchestrator.trident.netapp.io "trident" deleted
```

2. Der Trident-Operator kann mithilfe des Operator-Manifests gelöscht werden, das mit Ihrer Trident-Version verknüpft ist. Beispiel: `https://github.com/NetApp/trident/tree/stable/vXX.XX`

`/deploy/bundle.yaml` Wo `vXX.XX` Ist die Versionsnummer (zum Beispiel `v22.10`) Und `bundle.yaml` Ist der Name der Paket-YAML-Datei.

3. Löschen Sie die `tridentorchestrator` CRD.-

```
#Check to see if ``tridentorchestrators.trident.netapp.io`` CRD is
present and delete it.

kubectl get crd tridentorchestrators.trident.netapp.io

NAME                                        CREATED AT
tridentorchestrators.trident.netapp.io   2021-01-21T21:11:37Z

kubectl delete crd tridentorchestrators.trident.netapp.io

customresourcedefinition.apiextensions.k8s.io
"tridentorchestrators.trident.netapp.io" deleted
```

Astra Trident wurde deinstalliert.

4. Setzen Sie den Downgrade fort, indem Sie die gewünschte Version installieren. Befolgen Sie die Dokumentation für das gewünschte Release.

Downgrade mit Helm

Verwenden Sie zum Downgrade den `helm rollback` Befehl. Das folgende Beispiel zeigt:

```
helm rollback trident [revision #]
```

Downgrade bei Installation von Astra Trident mit `tridentctl`

Falls Sie Astra Trident mit installiert haben `tridentctl`, Der Downgrade-Prozess umfasst die folgenden Schritte. In dieser Folge werden Sie durch die Herabstufung von Astra Trident 21.07 auf 20.07 geleitet.

> Bevor Sie mit dem Downgrade beginnen, sollten Sie einen Schnappschuss Ihres Kubernetes-Clusters machen `etcd`. So können Sie den aktuellen Status der CRDs von Astra Trident sichern.

Schritte

1. Stellen Sie sicher, dass Trident mit installiert wird `tridentctl`. Wenn Sie sich nicht sicher sind, wie Astra Trident installiert ist, führen Sie diesen einfachen Test aus:
   a. Listen Sie die im Trident Namespace vorhandenen Pods auf.
   b. Ermitteln Sie die Version des Astra Trident in Ihrem Cluster. Sie können entweder verwenden `tridentctl` Oder werfen Sie einen Blick auf das Image, das in den Trident Pods verwendet wird.
   c. Wenn Sie * nicht sehen* a `tridentOrchestrator`, (Oder) a `tridentprovisioner`, (Oder) ein POD mit dem Namen `trident-operator-xxxxxxxxxx-xxxxx`, Astra Trident **ist installiert** mit

`tridentctl`.

2. Deinstallieren Sie Astra Trident mit den vorhandenen `tridentctl` Binär: In diesem Fall werden Sie mit dem 21.07 Binary deinstallieren.

```
tridentctl version -n trident
+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 21.07.0        | 21.07.0        |
+----------------+----------------+

tridentctl uninstall -n trident
INFO Deleted Trident deployment.
INFO Deleted Trident daemonset.
INFO Deleted Trident service.
INFO Deleted Trident secret.
INFO Deleted cluster role binding.
INFO Deleted cluster role.
INFO Deleted service account.
INFO Deleted pod security policy.
podSecurityPolicy=tridentpods
INFO The uninstaller did not delete Trident's namespace in case it is
going to be reused.
INFO Trident uninstallation succeeded.
```

3. Nachdem diese abgeschlossen ist, holen Sie sich die Trident-Binärdatei für die gewünschte Version (in diesem Beispiel, 20.07), und installieren Sie Astra Trident. Sie können benutzerdefinierte YAML für ein generieren "Benutzerdefinierte Installation" Wenn nötig.

```
cd 20.07/trident-installer/
./tridentctl install -n trident-ns
INFO Created installer service account.
serviceaccount=trident-installer
INFO Created installer cluster role.              clusterrole=trident-
installer
INFO Created installer cluster role binding.
clusterrolebinding=trident-installer
INFO Created installer configmap.                 configmap=trident-
installer
...
...
INFO Deleted installer cluster role binding.
INFO Deleted installer cluster role.
INFO Deleted installer service account.
```

Der Downgrade-Vorgang ist abgeschlossen.

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-Selbstheilung 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 "Die Dokumentation von redhat".

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.

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 (ANF) als Backend für Astra Trident konfigurieren. Sie können NFS- und SMB-Volumes über ein ANF-Backend anschließen.

- "Vorbereitung"
- "Konfigurationsoptionen und Beispiele"

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.
- Astra Trident unterstützt die Architektur von Windows ARM nicht.

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.

> 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".

Voraussetzungen für NFS und SMB Volumes

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/writ
e",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/snapshots/dele
te",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/subvolumes/rea
d",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/subvolumes/wri
te",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/subvolumes/del
ete",
```

```
"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/subvolumes/Get
Metadata/action",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/MountTargets/r
ead",
                    "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 Back-End-Konfigurationsoptionen für NFS und SMB für ANF und überprüfen Sie Konfigurationsbeispiele.

Astra Trident verwendet Ihre Backend-Konfiguration (Subnetz, virtuelles Netzwerk, Service Level und Standort), um ANF Volumes auf Kapazitäts-Pools zu erstellen, die am angeforderten Standort verfügbar sind und dem angeforderten Service Level und Subnetz entsprechen.

Astra Trident unterstützt keine manuellen QoS-Kapazitäts-Pools.

Back-End-Konfigurationsoptionen

ANF Back-Ends stellen diese Konfigurationsoptionen bereit.

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)

Parameter	Beschreibung	Standard
`netappAccounts`	Liste von NetApp Accounts zur Filterung erkannter Ressourcen	„[]“ (kein Filter)
`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. Angeben `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 `nfs`, `smb` Oder null. Einstellung auf null setzt standardmäßig auf NFS-Volumes.	`nfs`

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 eines PVC einen Fehler „Keine Kapazitätspools gefunden“ erhalten, ist es wahrscheinlich, dass Ihre App-Registrierung nicht die erforderlichen Berechtigungen und Ressourcen (Subnetz, virtuelles Netzwerk, Kapazitäts-Pool) zugeordnet hat. 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	<Ressourcengruppe>
NetApp Konto	<Resource Group>/<netapp Account>
Kapazitäts-Pool	<Resource Group>/<netapp Account>/<Capacity Pool>
Virtuelles Netzwerk	<Ressourcengruppe>/<virtuelles Netzwerk>
Subnetz	<Ressourcengruppe>/<virtuelles Netzwerk>/<Subnetz>

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 Ihre NetApp Konten, Kapazitäts-Pools und Subnetze, die an ANF am konfigurierten Speicherort delegiert wurden, und setzt zufällig neue Volumes auf einen dieser Pools und Subnetze. Weil `nasType` Wird weggelassen, das `nfs` Standard gilt und das Backend wird für NFS-Volumes bereitgestellt.

Diese Konfiguration eignet sich ideal, wenn Sie gerade mit ANF beginnen und die Dinge ausprobieren. In der Praxis möchten Sie jedoch zusätzliche Informationen für die Volumes bereitstellen, die Sie bereitstellen.

```
---
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 ANF delegierten Subnetze und legt ein neues Volume zufällig auf einen davon ab.

```
---
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: Virtuelle Pool-Konfiguration

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.

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

Parameter	Beschreibung	Standard
`apiRegion`	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. Der regionale Verkehr verursacht zusätzliche Kosten.	
`apiKey`	API-Schlüssel für das Google Cloud-Dienstkonto bei `netappcloudvolumes.admin` Rolle: Er enthält den JSON-formatierten Inhalt der privaten Schlüsseldatei eines Google Cloud-Dienstkontos (wortgetreu in die Back-End-Konfigurationsdatei kopiert).	
`proxyURL`	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. Bei einem HTTPS-Proxy wird die Zertifikatvalidierung übersprungen, um die Verwendung von selbstsignierten Zertifikaten im Proxyserver zu ermöglichen. Proxy-Server mit aktivierter Authentifizierung werden nicht unterstützt.	
`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`	Das CVS-Performance oder CVS Service-Level für neue Volumes. CVS-Performance Werte sind `standard`, `premium`, Oder `extreme`. CVS-Werte sind `standardsw` Oder `zoneredundantstandardsw`.	CVS-Performance ist der Standard. Der CVS-Standardwert ist „standardsw“.

Parameter	Beschreibung	Standard
`network`	Für Cloud Volumes Service Volumes verwendetes Google Cloud Netzwerk	„Standard“
`debugTraceFlags`	Fehler-Flags bei der Fehlerbehebung beheben. Beispiel: `\{"api":false, "method":true}`. Verwenden Sie dies nur, wenn Sie Fehler beheben und einen detaillierten Log Dump benötigen.	Null

`allowedTopologies`	Damit Sie regionsübergreifenden Zugriff ermöglichen, wird Ihre StorageClass-Definition für verwendet `allowedTopologies` Muss alle Regionen umfassen. Beispiel:
	`- 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 kommagetrennte 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`	Die Größe neuer Volumes. Die Mindestmenge von CVS-Performance beträgt 100 gib. CVS mindestens 1 gib.	Der Servicetyp CVS-Performance ist standardmäßig auf „100 gib“ eingestellt. CVS-Diensttyp setzt keine Standardeinstellung, erfordert jedoch mindestens 1 gib.

Beispiele für CVS-Performance-Diensttypen

Die folgenden Beispiele enthalten Beispielkonfigurationen 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-----
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/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
```

Beispiel 2: Service Level-Konfiguration

Dieses Beispiel stellt die Back-End-Konfigurationsoptionen dar, einschließlich Service Level und Volume-Standardeinstellungen.

```
---
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-----
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/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
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
```

Beispiel 3: Virtuelle Pool-Konfiguration

Dieses Beispiel verwendet `storage` Um virtuelle Pools und die zu konfigurieren `StorageClasses` Die sich auf sie beziehen. Siehe Definitionen der Storage-Klassen Um zu sehen, wie die Speicherklassen definiert wurden.

Hier werden für alle virtuellen Pools, die das festlegen, spezifische Standardeinstellungen festgelegt `snapshotReserve` Bei 5% und der `exportRule` Zu 0.0.0.0/0. Die virtuellen Pools werden im definiert `storage` Abschnitt. Jeder individuelle virtuelle Pool definiert seine eigenen `serviceLevel`, Und einige Pools überschreiben die Standardwerte. Virtuelle Pool-Labels wurden verwendet, um die Pools basierend auf zu differenzieren `performance` Und `protection`.

```
---
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-----
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
```

```
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/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.

Beispiel 1: Minimalkonfiguration

Dies ist die minimale Backend-Konfiguration mit `storageClass` Geben Sie den CVS-Diensttyp und den Standardwert an `standardsw` Service-Level:

```
---
version: 1
storageDriverName: gcp-cvs
projectNumber: '012345678901'
storageClass: software
apiRegion: us-east4
apiKey:
  type: service_account
  project_id: my-gcp-project
  private_key_id: "<id_value>"
  private_key: |
    -----BEGIN PRIVATE KEY-----
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/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
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-----
    MIIEvAIBADANBgkqhkiG9w0BAQEFAASCBKYwggSiAgEAAoIBAQDaT+Oui9FBAw19
    L1AGEkrYU5xd9K5NlO5jMkIFND5wCD+Nv+jd1GvtFRLaLK5RvXyF5wzvztmODNS+
    qtScpQ+5cFpQkuGtv9U9+N6qtuVYYO3b504Kp5CtqVPJCgMJaK2j8pZTIqUiMum/
    5/Y9oTbZrjAHSMgJm2nHzFq2X0rqVMaHghI6ATm4DOuWx8XGWKTGIPlc0qPqJlqS
    LLaWOH4VIZQZCAyW5IUp9CAmwqHgdG0uhFNfCgMmED6PBUvVLsLvcq86X+QSWR9k
    ETqElj/sGCenPF7ti1DhGBFafd9hPnxg9PZY29ArEZwY9G/ZjZQX7WPgs0VvxiNR
    DxZRC3GXAgMBAAECggEACn5c59bG/qnVEVI1CwMAalM5M2z09JFhlLlljKwntNPj
    Vilw2eTW2+UE7HbJru/S7KQgA5Dnn9kvCraEahPRuddUMrD0vG4kTl/IODV6uFuk
    Y0sZfbqd4jMUQ21smvGsqFzwloYWS5qzO1W83ivXH/HW/iqkmY2eW+EPRS/hwSSu
    SscR+SojI7PB0BWSJhlV4yqYf3vcD/D95el2CVHfRCkL85DKumeZ+yHEnpiXGZAE
    t8xSs4a5OOPm6NHhevCw2a/UQ95/foXNUR450HtbjieJo5o+FF6EYZQGfU2ZHZO8
    37FBKuaJkdGW5xqaI9TL7aqkGkFMF4F2qvOZM+vy8QKBgQD4oVuOkJDlhkTHP86W
    esFlw1kpWyJR9ZA7LI0g/rVpslnX+XdDq0WQf4umdLNau5hYEH9LU6ZSGs1Xk3/B
    NHwR6OXFuqEKNiu83d0zSlHhTy7PZpOZdj5a/vVvQfPDMz7OvsqLRd7YCAbdzuQ0
    +Ahq0Ztwvg0HQ64hdW0ukpYRRwKBgQDgyHj98oqswoYuIa+pP1yS0pPwLmjwKyNm
    /HayzCp+Qjiyy7Tzg8AUqlH1Ou83XbV428jvg7kDhO7PCCKFq+mMmfqHmTpb0Maq
    KpKnZg4ipsqP1yHNNEoRmcailXbwIhCLewMqMrggUiLOmCw4PscL5nK+4GKu2XE1
    jLqjWAZFMQKBgFHkQ9XXRAJ1kR3XpGHoGN890pZOkCVSrqju6aUef/5KYlFCt8ew
    F/+aIxM2iQSvmWQYOvVCnhuY/F2GFaQ7d0om3decuwI0CX/xy7PjHMkLXa2uaZs4
    WR17sLduj62RqXRLX0c0QkwBiNFyHbRcpdkZJQujbYMhBa+7j7SxT4BtAoGAWMWT
    UucocRXZm/pdvz9wteNH3YDWnJLMxm1KC06qMXbBoYrliY4sm3ywJWMC+iCd/H8A
    Gecxd/xVu5mA2L2N3KMq18Zhz8Th0G5DwKyDRJgOQ0Q46yuNXOoYEjlo4Wjyk8Me
    +tlQ8iK98E0UmZnhTgfSpSNElbz2AqnzQ3MN9uECgYAqdvdVPnKGfvdtZ2DjyMoJ
    E89UIC41WjjJGmHsd8W65+3X0RwMzKMT6aZc5tK9J5dHvmWIETnbM+lTImdBBFga
    NWOC6f3r2xbGXHhaWSl+nobpTuvlo56ZRJVvVk7lFMsiddzMuHH8pxfgNJemwA4P
    ThDHCejv035NNV6KyoO0tA==
    -----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

Erfahren Sie, wie Sie mit Ihrer Astra Trident Installation ein Element Backend erstellen und verwenden.

Was Sie benötigen

- 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".

Was Sie wissen müssen

Der `solidfire-san` Der Storage-Treiber unterstützt beide Volume-Modi: Datei 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	Kein Dateisystem. Rohes Blockgerät.
`solidfire-san`	ISCSI	Block-Storage	RWO, ROX, RWX	Kein Dateisystem. Rohes Blockgerät.
`solidfire-san`	ISCSI	Dateisystem	RWO, ROX	`xfs`, `ext3`, `ext4`
`solidfire-san`	ISCSI	Dateisystem	RWO, ROX	`xfs`, `ext3`, `ext4`

Astra Trident verwendet CHAP, wenn es als erweiterte CSI-Bereitstellung funktioniert. Wenn Sie CHAP verwenden (das ist die Standardeinstellung für CSI), ist keine weitere Vorbereitung erforderlich. Es wird empfohlen, das explizit festzulegen `UseCHAP` Option zur Verwendung von CHAP mit nicht-CSI Trident. Anderenfalls siehe "Hier".

Volume-Zugriffsgruppen werden nur vom herkömmlichen, nicht-CSI-Framework für Astra Trident unterstützt. Bei der Konfiguration für die Verwendung im CSI-Modus verwendet Astra Trident CHAP.

Wenn keine `AccessGroups` Oder `UseCHAP` Sind festgelegt, gilt eines der folgenden Regeln:

- Wenn die Standardeinstellung `trident` Zugriffsgruppe wird erkannt, Zugriffsgruppen werden verwendet.
- Wenn keine Zugriffsgruppe erkannt wird und die Kubernetes-Version 1.7 oder höher ist, wird CHAP verwendet.

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.	„“

Parameter	Beschreibung	Standard
`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, um iSCSI zu authentifizieren	Richtig
`AccessGroups`	Liste der zu verwendenden Zugriffsgruppen-IDs	Findet die ID einer Zugriffsgruppe namens „Dreizack“
`Types`	QoS-Spezifikationen	
`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 legt ein Teil des Speicherpools seinen 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"

Konfigurieren Sie ein Backend mit ONTAP-SAN-Treibern

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

- "Vorbereitung"
- "Konfiguration und Beispiele"

> 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.
>
> - Sie müssen verwenden `ontap-nas` Für produktive Workloads, die Datensicherung, Disaster Recovery und Mobilität erfordern.
> - Nutzung `ontap-san-economy` Nach einer voraussichtlichen Volume-Nutzung ist zu erwarten, dass sie wesentlich höher ist, als das von ONTAP unterstützt wird.
> - Nutzung `ontap-nas-economy` Nur in dem eine zu erwartende Volume-Nutzung größer als die von ONTAP wird, und `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.

Obwohl es möglich ist, eine restriktivere Rolle innerhalb ONTAP, dass ein Trident-Treiber verwenden kann, wir nicht empfehlen es. 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

Erfahren Sie, wie Sie ein ONTAP-Back-End mit ONTAP-SAN-Treibern vorbereiten. 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 "Hier" Entnehmen.

Authentifizierung

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: BeispieleBackend storageDriverName: ontap-san ManagementLIF: 10.0.0.1 svm: svm_nfs Benutzername: Vsadmin Passwort: Passwort

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 vertrauenswürdiges 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...Vaaalllluuuueeee",
"clientPrivateKey": "LS0tFaKE...0VaLuES0tLS0K",
"trustedCACertificate": "QNFinfO...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.

Geben Sie Initiatorgruppen an

Astra Trident verwendet Initiatorgruppen, um den Zugriff auf die Volumes (LUNs) zu steuern, die er bereitstellt. Administratoren verfügen über zwei Optionen, wenn es um das Angeben von Initiatorgruppen für Back-Ends geht:

- Astra Trident kann automatisch eine igroup pro Backend erstellen und managen. Wenn `igroupName` Ist nicht in der Backend-Definition enthalten, erstellt Astra Trident eine igroup mit dem Namen `trident-<backend-UUID>` Auf der SVM. So wird sichergestellt, dass jedes Backend über eine dedizierte iGroup verfügt und das automatisierte Hinzufügen/Löschen von Kubernetes Node-IQNs behandelt.

- Alternativ können auch vorab erstellte Initiatorgruppen in einer Backend-Definition bereitgestellt werden. Dies kann mit dem erfolgen `igroupName` Konfigurationsparameter. Astra Trident fügt der bereits vorhandenen iGroup Kubernetes-Node-IQNs hinzu/löschen.

Für Back-Ends mit `igroupName` Definiert, das `igroupName` Kann mit einem gelöscht werden `tridentctl backend update` Astra Trident ist die Auto-Handle-Initiatorgruppen. Dadurch wird der Zugriff auf Volumes nicht unterbrochen, die bereits an Workloads angeschlossen sind. Künftige Verbindungen werden mit der von der igroup Astra Trident erstellten iGroup behandelt.

> Die Einwidmung einer Initiatorgruppe für jede einzelne Instanz des Astra Trident ist eine Best Practice, die sowohl dem Kubernetes-Administrator als auch dem Storage-Administrator von Vorteil ist. CSI Trident automatisiert das Hinzufügen und Entfernen von Cluster Node-IQNs zur igroup und vereinfacht das Management enorm. Wenn in Kubernetes-Umgebungen dieselben SVMs verwendet werden (und Astra Trident-Installationen), stellt die Verwendung einer dedizierten igroup sicher, dass Änderungen an einem Kubernetes-Cluster keinen Einfluss auf Initiatorgruppen haben, die anderen zugeordnet sind. Darüber hinaus ist es wichtig, dass jeder Node im Kubernetes Cluster über einen eindeutigen IQN verfügt. Wie oben erwähnt, übernimmt Astra Trident automatisch das Hinzufügen und Entfernen von IQNs. Die Wiederverwendung von IQNs über Hosts kann zu unerwünschten Szenarien führen, in denen Hosts sich gegenseitig irren und der Zugriff auf LUNs verweigert wird.

Wenn Astra Trident als CSI-Bereitstellung konfiguriert ist, werden Kubernetes-Node-IQNs automatisch der Initiatorgruppe hinzugefügt/entfernt. Wenn Nodes zu einem Kubernetes-Cluster hinzugefügt werden, `trident-csi` DemonSet setzt einen POD ein (`trident-csi-xxxxx` In Versionen vor 23.01 oder `trident-node<operating system>-xxxx` Ab 23.01) auf den neu hinzugefügten Knoten und registriert die neuen Knoten, an die es Volumes hinzufügen kann. Node-IQNs werden ebenfalls zur iGroup des Backend hinzugefügt. Eine ähnliche Reihe von Schritten behandelt das Entfernen von IQNs, wenn Nodes aus Kubernetes abgesperrt, entleert und gelöscht werden.

Wenn Astra Trident nicht als CSI-Bereitstellung ausgeführt wird, muss die Initiatorgruppe manuell aktualisiert werden, um die iSCSI-IQNs von jedem Worker-Node im Kubernetes-Cluster zu enthalten. IQNs von Nodes, die dem Kubernetes-Cluster beitreten, müssen zur Initiatorgruppe hinzugefügt werden. Ebenso müssen IQNs von Nodes, die aus dem Kubernetes-Cluster entfernt werden, aus der Initiatorgruppe entfernt werden.

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 legt den Benutzernamen und die Schlüssel 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
igroupName: trident
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 Standardsicherheitstyp des SVM keine (standardmäßig eingestellt) ist **und** gibt es keine bereits vorhandenen LUNs im Volume, setzt Astra Trident den Standardsicherheitstyp 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 LUNs, die bereits auf der SVM vorhanden sind, nicht beschränkt ist.
- Konfigurieren des CHAP-Initiators und des Ziel-Usernamens und der Schlüssel; diese Optionen müssen in der Back-End-Konfiguration angegeben werden (siehe oben).
- Verwalten des Hinzufügung von Initiatoren zu dem `igroupName` Gegeben im Backend. Wenn die Angabe nicht festgelegt ist, wird standardmäßig auf diese Option gesetzt `trident`.

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",
    "igroupName": "trident",
    "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 mit Ihrer Installation von Astra Trident ONTAP SAN-Treiber erstellen und verwenden. Dieser Abschnitt enthält Beispiele für die Back-End-Konfiguration und Details zur 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`	IP-Adresse eines Clusters oder SVM-Management-LIF für nahtlose MetroCluster-Umschaltung müssen Sie eine SVM-Management-LIF angeben. Es kann ein vollständig qualifizierter Domänenname (FQDN) angegeben werden. Kann so eingestellt werden, dass IPv6-Adressen verwendet werden, wenn Astra Trident mit installiert wurde `--use-ipv6` Flagge. IPv6-Adressen müssen in eckigen Klammern definiert werden, z. B. [28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555].	„10.0.0.1“, „[2001:1234:abcd::fefe]“
`dataLIF`	IP-Adresse des LIF-Protokolls. **Nicht für iSCSI angeben.** Astra Trident verwendet "ONTAP selektive LUN-Zuordnung" Um die iSCI LIFs zu ermitteln, die für die Einrichtung einer Multi-Path-Sitzung erforderlich sind. Wenn eine Warnung erzeugt wird `dataLIF` Ist explizit definiert.	Abgeleitet von SVM

Parameter	Beschreibung	Standard
`useCHAP`	Verwenden Sie CHAP, um iSCSI für ONTAP-SAN-Treiber zu authentifizieren [Boolesch]. 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.	Falsch
`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

Parameter	Beschreibung	Standard
`igroupName`	Der Name der Initiatorgruppe für die zu verwendenden SAN Volumes. Siehe Finden Sie weitere Informationen.	„Trident-<Backend-UUID>“
`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.	„Dreizack“
`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)
`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.	„“ (nicht standardmäßig durchgesetzt)
`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, „Methode“:true} Verwenden Sie nur, wenn Sie eine Fehlerbehebung durchführen und einen detaillierten Logdump benötigen.	Null

Parameter	Beschreibung	Standard
`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 `true`, 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 `ontap` Applikation. Dies wird durch die vordefinierte zufrieden `vsadmin` Und `cluster-admin` Rollen: `useREST` Wird mit MetroCluster nicht unterstützt.	Falsch

Details zu `igroupName`

`igroupName` Kann auf eine Initiatorgruppe festgelegt werden, die bereits auf dem ONTAP Cluster erstellt wurde. Wenn nicht angegeben, erstellt Astra Trident automatisch eine igroup mit dem Namen `trident-<backend-UUID>`.

Bei Bereitstellung eines vordefinierten igroupName empfehlen wir die Verwendung einer Initiatorgruppe pro Kubernetes Cluster, sofern die SVM zwischen Umgebungen gemeinsam genutzt werden soll. Dies ist notwendig, damit Astra Trident automatisch IQN-Ergänzungen und -Löschungen pflegt.

- `igroupName` Kann aktualisiert werden, um auf eine neue igroup zu verweisen, die auf der SVM außerhalb des Astra Trident erstellt und gemanagt wird.
- `igroupName` Kann weggelassen werden. In diesem Fall wird Astra Trident eine igroup mit dem Namen erstellen und verwalten `trident-<backend-UUID>` Automatisch

In beiden Fällen können Sie weiterhin auf Volume-Anhänge zugreifen. Zukünftige Volume-Anhänge verwenden die aktualisierte Initiatorgruppe. Dieses Update wird den Zugriff auf Volumes im Backend nicht unterbrechen.

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`	Space Reservation Mode; „none“ (Thin) oder „Volume“ (Thick)	„Keine“

Parameter	Beschreibung	Standard
`snapshotPolicy`	Die Snapshot-Richtlinie zu verwenden	„Keine“
`qosPolicy`	QoS-Richtliniengruppe zur Zuweisung für erstellte Volumes Wählen Sie eine der qosPolicy oder adaptiveQosPolicy pro Storage Pool/Backend. 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.	„“
`adaptiveQosPolicy`	Adaptive QoS-Richtliniengruppe mit Zuordnung für erstellte Volumes Wählen Sie eine der qosPolicy oder adaptiveQosPolicy pro Storage Pool/Backend	„“
`snapshotReserve`	Prozentsatz des für Snapshots reservierten Volumens „0“	Wenn `snapshotPolicy` Ist „keine“, sonst „“
`splitOnClone`	Teilen Sie einen Klon bei der Erstellung von seinem übergeordneten Objekt auf	„Falsch“
`encryption`	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. 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“
`luksEncryption`	Aktivieren Sie die LUKS-Verschlüsselung. Siehe "Linux Unified Key Setup (LUKS) verwenden".	„“
`securityStyle`	Sicherheitstyp für neue Volumes	`unix`

Parameter	Beschreibung	Standard
`tieringPolicy`	Tiering-Richtlinie zur Verwendung von „keiner“	„Nur Snapshot“ für eine ONTAP 9.5 SVM-DR-Konfiguration

Beispiele für die Volume-Bereitstellung

Hier ist ein Beispiel mit definierten Standardeinstellungen:

```
---
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
igroupName: custom
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:

```
Total volume size = [(PVC requested size) / (1 - (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:

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, empfiehlt es sich, DNS-Namen für LIFs anstelle von IP-Adressen anzugeben.

`ontap-san` Treiber mit zertifikatbasierter 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
storageDriverName: ontap-san
backendName: DefaultSANBackend
managementLIF: 10.0.0.1
svm: svm_iscsi
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSd6cNwxyz
igroupName: trident
clientCertificate: ZXR0ZXJwYXB...ICMgJ3BhcGVyc2
clientPrivateKey: vciwKIyAgZG...0cnksIGRlc2NyaX
trustedCACertificate: zcyBbaG...b3Igb3duIGNsYXNz
```

`ontap-san` Treiber mit bidirektionalem CHAP

Dies ist ein minimales Beispiel für die Back-End-Konfiguration. Mit dieser Grundkonfiguration wird ein erstellt

`ontap-san` Back-End mit `useCHAP` Auf einstellen `true`.

```
---
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
igroupName: trident
username: vsadmin
password: password
```

`ontap-san-economy` **Treiber**

```
---
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
igroupName: trident
username: vsadmin
password: password
```

Beispiele für Back-Ends mit virtuellen Pools

In der unten gezeigten Beispiel-Back-End-Definitionsdatei werden bestimmte Standardeinstellungen für alle Storage Pools festgelegt, z. B. `spaceReserve` Bei keiner, `spaceAllocation` Bei false, und `encryption` Bei false. Die virtuellen Pools werden im Abschnitt Speicher definiert.

Astra Trident setzt Provisioning-Labels im Bereich „Comments“. 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 diesem Beispiel legt ein Teil des Speicherpools seine eigenen fest `spaceReserve`, `spaceAllocation`,

und `encryption` Werte und einige Pools überschreiben die oben festgelegten Standardwerte.

```
---
version: 1
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_iscsi
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSd6cNwxyz
igroupName: trident
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'
```

Hier ist ein iSCSI-Beispiel für das `ontap-san-economy` Treiber:

```
---
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
igroupName: trident
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'
```

Back-Ends StorageClasses zuordnen

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 (`protection-gold`) Wird dem ersten, zweiten virtuellen Pool im zugeordnet `ontap-nas-flexgroup` Back-End und der erste virtuelle Pool im `ontap-san` Back-End: Dies sind die einzigen Pools, die Schutz auf Goldebene bieten.
- Die zweite StorageClass (`protection-not-gold`) Wird dem dritten, vierten virtuellen Pool in zugeordnet `ontap-nas-flexgroup` Back-End und der zweite, dritte virtuelle Pool in `ontap-san` Back-End: Dies sind die einzigen Pools, die Schutz Level nicht Gold bieten.
- Die dritte StorageClass (`app-mysqldb`) Wird dem vierten virtuellen Pool in zugeordnet `ontap-nas` Back-End und der dritte virtuelle Pool in `ontap-san-economy` Back-End: Dies sind die einzigen Pools, die eine Storage-Pool-Konfiguration für die mysqldb-Typ-App bieten.
- Die vierte StorageClass (`protection-silver-creditpoints-20k`) Wird dem dritten virtuellen Pool in zugeordnet `ontap-nas-flexgroup` Back-End und der zweite virtuelle Pool in `ontap-san` Back-End: Dies sind die einzigen Pools, die Gold-Level-Schutz mit 20000 Kreditpunkten bieten.
- Die fünfte StorageClass (`creditpoints-5k`) Wird dem zweiten virtuellen Pool in zugeordnet `ontap-nas-economy` Back-End und der dritte virtuelle Pool in `ontap-san` Back-End: Dies sind die einzigen Poolangebote mit 5000 Kreditpunkten.

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: protection-gold
provisioner: netapp.io/trident
parameters:
  selector: "protection=gold"
  fsType: "ext4"
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-not-gold
provisioner: netapp.io/trident
parameters:
  selector: "protection!=gold"
  fsType: "ext4"
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: app-mysqldb
provisioner: netapp.io/trident
parameters:
  selector: "app=mysqldb"
  fsType: "ext4"
---
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"
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: creditpoints-5k
provisioner: netapp.io/trident
parameters:
  selector: "creditpoints=5000"
  fsType: "ext4"
```

Konfigurieren Sie ein ONTAP-NAS-Back-End

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

- "Vorbereitung"
- "Konfiguration und Beispiele"

> 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.
>
> - Sie müssen verwenden `ontap-nas` Für produktive Workloads, die Datensicherung, Disaster Recovery und Mobilität erfordern.
> - Nutzung `ontap-san-economy` Nach einer voraussichtlichen Volume-Nutzung ist zu erwarten, dass sie wesentlich höher ist, als das von ONTAP unterstützt wird.
> - Nutzung `ontap-nas-economy` Nur in dem eine zu erwartende Volume-Nutzung größer als die von ONTAP wird, und `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.

Obwohl es möglich ist, eine restriktivere Rolle innerhalb ONTAP, dass ein Trident-Treiber verwenden kann, wir nicht empfehlen es. 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

Erfahren Sie, wie Sie ein ONTAP-Back-End mit ONTAP-NAS-Treibern vorbereiten. Für alle ONTAP Back-Ends benötigt Astra Trident mindestens ein Aggregat, das der SVM zugewiesen ist.

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 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.

Authentifizierung

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 vertrauenswürdiges 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 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-updated.json
{
"version": 1,
"storageDriverName": "ontap-nas",
"backendName": "NasBackend",
"managementLIF": "1.2.3.4",
"dataLIF": "1.2.3.8",
"svm": "vserver_test",
"clientCertificate": "Faaaakkkkeeee...Vaaalllluuuueeee",
"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

Mit der Version 20.04 von CSI Trident können Exportrichtlinien für ONTAP-Back-Ends dynamisch gemanagt werden. 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.

Das dynamische Management der Exportrichtlinien steht nur für CSI Trident zur Verfügung. Es ist wichtig sicherzustellen, dass die Worker Nodes nicht NATed werden.

Beispiel

Es müssen zwei Konfigurationsoptionen verwendet werden. Hier ist ein 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 immer der von NetApp empfohlenen Best Practice, eine SVM für Astra Trident einzurichten.

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 globellem 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>
      fileSystemType: 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 in 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 Backend benannt ist und Volumes, die auf dem Backend vorhanden sind, verwenden die neu erstellte Exportrichtlinie, wenn sie erneut 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.

ONTAP NAS-Konfigurationsoptionen und -Beispiele

Erfahren Sie, wie Sie mit Ihrer Installation von Astra Trident ONTAP NAS-Treiber erstellen und verwenden. Dieser Abschnitt enthält Beispiele für die Back-End-Konfiguration und Details zur 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`	IP-Adresse eines Clusters oder SVM-Management-LIF für nahtlose MetroCluster-Umschaltung müssen Sie eine SVM-Management-LIF angeben. Es kann ein vollständig qualifizierter Domänenname (FQDN) angegeben werden. Kann so eingestellt werden, dass IPv6-Adressen verwendet werden, wenn Astra Trident mit installiert wurde `--use-ipv6` Flagge. IPv6-Adressen müssen in eckigen Klammern definiert werden, z. B. [28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555].	„10.0.0.1“, „[2001:1234:abcd::fefe]“

Parameter	Beschreibung	Standard
`dataLIF`	IP-Adresse des LIF-Protokolls. Wir empfehlen Ihnen, 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. Kann nach der Anfangseinstellung geändert werden. Siehe . Kann so eingestellt werden, dass IPv6-Adressen verwendet werden, wenn Astra Trident mit installiert wurde `--use -ipv6` Flagge. IPv6-Adressen müssen in eckigen Klammern definiert werden, z. B. [28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555].	Angegebene Adresse oder abgeleitet von SVM, falls nicht angegeben (nicht empfohlen)
`autoExportPolicy`	Aktivieren Sie die automatische Erstellung von Exportrichtlinien und aktualisieren Sie [Boolean]. Verwenden der `autoExportPolicy` Und `autoExportCIDRs` Optionen: Astra Trident kann Exportrichtlinien automatisch verwalten.	Falsch
`autoExportCIDRs`	Liste der CIDRs, um die Kubernetes-Knoten-IPs gegen Wann zu filtern `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	„“

Parameter	Beschreibung	Standard
`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	
`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 nicht aktualisiert werden, nachdem Sie sie festgelegt haben	„Dreizack“
`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.	„“ (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.	„“ (nicht standardmäßig durchgesetzt)
`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, „Methode“:true} Verwenden Sie nicht `debugTraceFlags` Es sei denn, Sie beheben Fehler und benötigen einen detaillierten Log Dump.	Null

Parameter	Beschreibung	Standard
`nfsMountOptions`	Kommagetrennte Liste von NFS-Mount-Optionen. 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. 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.	„“
`qtreesPerFlexvol`	Maximale Ques pro FlexVol, muss im Bereich [50, 300] liegen	„200“
`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 `true`, 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 `ontap` Applikation. Dies wird durch die vordefinierte zufrieden `vsadmin` Und `cluster-admin` Rollen: `useREST` Wird mit MetroCluster nicht unterstützt.	Falsch

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`	Space Reservation Mode; „none“ (Thin) oder „Volume“ (Thick)	„Keine“

Parameter	Beschreibung	Standard
`snapshotPolicy`	Die Snapshot-Richtlinie zu verwenden	„Keine“
`qosPolicy`	QoS-Richtliniengruppe zur Zuweisung für erstellte Volumes Wählen Sie eine der qosPolicy oder adaptiveQosPolicy pro Storage Pool/Backend	„“
`adaptiveQosPolicy`	Adaptive QoS-Richtliniengruppe mit Zuordnung für erstellte Volumes Wählen Sie eine der qosPolicy oder adaptiveQosPolicy pro Storage Pool/Backend. Nicht unterstützt durch ontap-nas-Ökonomie	„“
`snapshotReserve`	Prozentsatz des für Snapshots reservierten Volumens „0“	Wenn `snapshotPolicy` Ist „keine“, sonst „“
`splitOnClone`	Teilen Sie einen Klon bei der Erstellung von seinem übergeordneten Objekt auf	„Falsch“
`encryption`	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. 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“
`tieringPolicy`	Tiering-Richtlinie zur Verwendung von „keiner“	„Nur Snapshot“ für eine ONTAP 9.5 SVM-DR-Konfiguration
`unixPermissions`	Modus für neue Volumes	„777“ für NFS Volumes; leer (nicht zutreffend) für SMB Volumes
`snapshotDir`	Steuert die Sichtbarkeit des `.snapshot` Verzeichnis	„Falsch“
`exportPolicy`	Zu verwendende Exportrichtlinie	„Standard“
`securityStyle`	Sicherheitstyp für neue Volumes. NFS unterstützt `mixed` Und `unix` Sicherheitsstile. SMB unterstützt `mixed` Und `ntfs` Sicherheitsstile.	NFS-Standard ist `unix`. SMB-Standard ist `ntfs`.

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:

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.

Beispiele

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.

Standardoptionen ein <code>ontap-nas-economy</code>

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

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_
```

Automatische Exportrichtlinie

Diese Beispiele zeigen Ihnen, 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.

ontap-nas-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
```

<code>ontap-nas-flexgroup</code> Treiber

```
---
version: 1
storageDriverName: ontap-nas-flexgroup
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
labels:
  k8scluster: test-cluster-east-1b
  backend: test1-ontap-cluster
svm: svm_nfs
username: vsadmin
password: password
```

Verwenden von 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
```

`ontap-nas-economy` Treiber

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

`ontap-nas` Treiber für Amazon FSX für ONTAP mithilfe von SMB Volumes

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

Beispiele für Back-Ends mit virtuellen Pools

In der unten gezeigten Beispiel-Back-End-Definitionsdatei werden bestimmte Standardeinstellungen für alle Storage Pools festgelegt, z. B. `spaceReserve` Bei keiner, `spaceAllocation` Bei false, und `encryption` Bei false. Die virtuellen Pools werden im Abschnitt Speicher definiert.

Astra Trident setzt Provisioning-Labels im Bereich „Comments“. Kommentare wurden 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 diesem Beispiel legt ein Teil des Speicherpools seine eigenen fest `spaceReserve`, `spaceAllocation`, und `encryption` Werte und einige Pools überschreiben die oben festgelegten Standardwerte.

<code>ontap-nas</code> Treiber

```
---
version: 1
storageDriverName: ontap-nas
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
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:
    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'
```

<code>ontap-nas-flexgroup</code> Treiber

```
---
version: 1
storageDriverName: ontap-nas-flexgroup
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
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'
```

<code>ontap-nas-economy</code> Treiber

```
---
version: 1
storageDriverName: ontap-nas-economy
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
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:
    department: legal
    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'
```

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-Ends StorageClasses zuordnen

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 (`protection-gold`) Wird dem ersten, zweiten virtuellen Pool im zugeordnet `ontap-nas-flexgroup` Back-End und der erste virtuelle Pool im `ontap-san` Back-End: Dies sind die einzigen Pools, die Schutz auf Goldebene bieten.
- Die zweite StorageClass (`protection-not-gold`) Wird dem dritten, vierten virtuellen Pool in zugeordnet `ontap-nas-flexgroup` Back-End und der zweite, dritte virtuelle Pool in `ontap-san` Back-End: Dies sind die einzigen Pools, die Schutz Level nicht Gold bieten.
- Die dritte StorageClass (`app-mysqldb`) Wird dem vierten virtuellen Pool in zugeordnet `ontap-nas` Back-End und der dritte virtuelle Pool in `ontap-san-economy` Back-End: Dies sind die einzigen Pools, die eine Storage-Pool-Konfiguration für die mysqldb-Typ-App bieten.
- Die vierte StorageClass (`protection-silver-creditpoints-20k`) Wird dem dritten virtuellen Pool in zugeordnet `ontap-nas-flexgroup` Back-End und der zweite virtuelle Pool in `ontap-san` Back-End: Dies sind die einzigen Pools, die Gold-Level-Schutz mit 20000 Kreditpunkten bieten.
- Die fünfte StorageClass (`creditpoints-5k`) Wird dem zweiten virtuellen Pool in zugeordnet `ontap-nas-economy` Back-End und der dritte virtuelle Pool in `ontap-san` Back-End: Dies sind die einzigen Poolangebote mit 5000 Kreditpunkten.

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: protection-gold
provisioner: netapp.io/trident
parameters:
  selector: "protection=gold"
  fsType: "ext4"
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-not-gold
provisioner: netapp.io/trident
parameters:
  selector: "protection!=gold"
  fsType: "ext4"
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: app-mysqldb
provisioner: netapp.io/trident
parameters:
  selector: "app=mysqldb"
  fsType: "ext4"
---
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"
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: creditpoints-5k
provisioner: netapp.io/trident
parameters:
  selector: "creditpoints=5000"
  fsType: "ext4"
```

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.

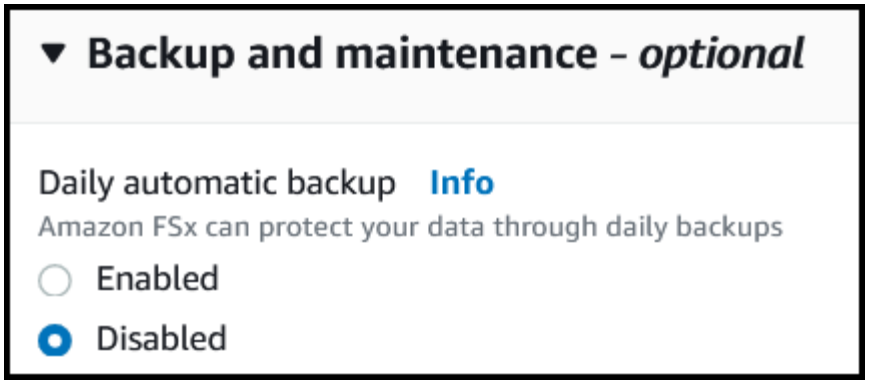
Ein Dateisystem ist die primäre Ressource in Amazon FSX, analog zu einem ONTAP-Cluster vor Ort. Innerhalb jeder SVM können Sie ein oder mehrere Volumes erstellen, bei denen es sich um Daten-Container handelt, die die Dateien und Ordner im Filesystem speichern. Amazon FSX für NetApp ONTAP wird Data ONTAP 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 nutzt "FabricPool" zum Management von Storage-Tiers. 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.
  - Astra Trident unterstützt die Architektur von Windows ARM nicht.
- 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.



Treiber

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 "ONTAP-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.

Bevor Sie beginnen

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 für NetApp ONTAP Filesystem und 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.

Zusätzliche Anforderungen für SMB Volumes

- 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.

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-XXXXXXXXXXXXXXXXX.fs-XXXXXXXXXXXXXXXXX.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-XXXXXXXXXXXXXXXXX.fs-
XXXXXXXXXXXXXXXXX.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"

Ergebnisse

Nach der Bereitstellung können Sie ein erstellen "Storage-Klasse, Volumes bereitstellen und das Volume in einem POD mounten".

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.

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`	Name der SMB-Freigabe, die mithilfe der Microsoft Management Console für freigegebene Ordner erstellt wurde. Zum Beispiel „smb-Share“. **Erforderlich für SMB Volumes.**	`smb-share`

Parameter	Beschreibung	Beispiel
`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
`managementLIF`	IP-Adresse eines Clusters oder SVM-Management-LIF für nahtlose MetroCluster-Umschaltung müssen Sie eine SVM-Management-LIF angeben. Es kann ein vollständig qualifizierter Domänenname (FQDN) angegeben werden. Kann so eingestellt werden, dass IPv6-Adressen verwendet werden, wenn Astra Trident mit installiert wurde `--use-ipv6` Flagge. IPv6-Adressen müssen in eckigen Klammern definiert werden, z. B. [28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555].	„10.0.0.1“, „[2001:1234:abcd::fefe]“

Parameter	Beschreibung	Beispiel
`dataLIF`	IP-Adresse des LIF-Protokolls. **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 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. Kann nach der Anfangseinstellung geändert werden. Siehe . **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. Kann so eingestellt werden, dass IPv6-Adressen verwendet werden, wenn Astra Trident mit installiert wurde `--use-ipv6` Flagge. IPv6-Adressen müssen in eckigen Klammern definiert werden, z. B. [28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555].	
`autoExportPolicy`	Aktivieren Sie die automatische Erstellung von Exportrichtlinien und aktualisieren Sie [Boolean]. Verwenden der `autoExportPolicy` Und `autoExportCIDRs` Optionen: Astra Trident kann Exportrichtlinien automatisch verwalten.	„Falsch“
`autoExportCIDRs`	Liste der CIDRs, um die Kubernetes-Knoten-IPs gegen Wann zu filtern `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	„“

Parameter	Beschreibung	Beispiel
`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.
`igroupName`	Der Name der Initiatorgruppe für die zu verwendenden SAN Volumes. Siehe .	„Trident-<Backend-UUID>“
`storagePrefix`	Das Präfix wird beim Bereitstellen neuer Volumes in der SVM verwendet. Kann nach der Erstellung nicht geändert werden. Um diesen Parameter zu aktualisieren, müssen Sie ein neues Backend erstellen.	„Dreizack“
`limitAggregateUsage`	**Für Amazon FSX nicht für NetApp ONTAP angeben.** den angegebenen `fsxadmin` Und `vsadmin` Enthalten Sie nicht die erforderlichen Berechtigungen, um die Aggregatnutzung abzurufen und sie mit Astra Trident zu begrenzen.	Verwenden Sie ihn nicht.

Parameter	Beschreibung	Beispiel
`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.	„“ (nicht standardmäßig durchgesetzt)
`lunsPerFlexvol`	Die maximale Anzahl an LUNs pro FlexVol muss im Bereich [50, 200] liegen. Nur SAN	„100“
`debugTraceFlags`	Fehler-Flags bei der Fehlerbehebung beheben. Beispiel: { „API“:false, „Methode“:true} Verwenden Sie nicht `debugTraceFlags` Es sei denn, Sie beheben Fehler und benötigen einen detaillierten Log Dump.	Null
`nfsMountOptions`	Kommagetrennte Liste von NFS-Mount-Optionen. 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. 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.	„“
`nasType`	Konfiguration der Erstellung von NFS- oder SMB-Volumes Die Optionen lauten `nfs`, `smb`, Oder Null. **Muss auf eingestellt sein `smb` Für SMB-Volumes.** Einstellung auf null setzt standardmäßig auf NFS-Volumes.	nfs
`qtreesPerFlexvol`	Maximale Ques pro FlexVol, muss im Bereich [50, 300] liegen	„200“

Parameter	Beschreibung	Beispiel
`smbShare`	Name der SMB-Freigabe, die mithilfe der Microsoft Management Console für freigegebene Ordner erstellt wurde. **Erforderlich für SMB Volumes.**	smb-Share
`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 `true`, 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 `ontap` Applikation. Dies wird durch die vordefinierte zufrieden `vsadmin` Und `cluster-admin` Rollen:	„Falsch“

Details zu `igroupName`

`igroupName` Kann auf eine Initiatorgruppe festgelegt werden, die bereits auf dem ONTAP Cluster erstellt wurde. Wenn nicht angegeben, erstellt Astra Trident automatisch eine igroup mit dem Namen `trident-<backend-UUID>`.

Bei Bereitstellung eines vordefinierten igroupName empfehlen wir die Verwendung einer Initiatorgruppe pro Kubernetes Cluster, sofern die SVM zwischen Umgebungen gemeinsam genutzt werden soll. Dies ist notwendig, damit Astra Trident automatisch IQN-Ergänzungen und -Löschungen pflegt.

- `igroupName` Kann aktualisiert werden, um auf eine neue igroup zu verweisen, die auf der SVM außerhalb des Astra Trident erstellt und gemanagt wird.
- `igroupName` Kann weggelassen werden. In diesem Fall wird Astra Trident eine igroup mit dem Namen erstellen und verwalten `trident-<backend-UUID>` Automatisch

In beiden Fällen können Sie weiterhin auf Volume-Anhänge zugreifen. Zukünftige Volume-Anhänge verwenden die aktualisierte Initiatorgruppe. Dieses Update wird den Zugriff auf Volumes im Backend nicht unterbrechen.

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
`spaceAllocation`	Speicherplatzzuweisung für LUNs	„Wahr“
`spaceReserve`	Space Reservation Mode; „none“ (Thin) oder „Volume“ (Thick)	„Keine“
`snapshotPolicy`	Die Snapshot-Richtlinie zu verwenden	„Keine“
`qosPolicy`	QoS-Richtliniengruppe zur Zuweisung für erstellte Volumes Wählen Sie eine der qosPolicy oder adaptiveQosPolicy pro Storage-Pool oder Backend. 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.	„“
`adaptiveQosPolicy`	Adaptive QoS-Richtliniengruppe mit Zuordnung für erstellte Volumes Wählen Sie eine der qosPolicy oder adaptiveQosPolicy pro Storage-Pool oder Backend. Nicht unterstützt durch ontap-nas-Ökonomie	„“
`snapshotReserve`	Prozentsatz des für Snapshots reservierten Volumens „0“	Wenn `snapshotPolicy` Ist „keine“, sonst „“

Parameter	Beschreibung	Standard
`splitOnClone`	Teilen Sie einen Klon bei der Erstellung von seinem übergeordneten Objekt auf	„Falsch“
`encryption`	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. 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“
`luksEncryption`	Aktivieren Sie die LUKS-Verschlüsselung. Siehe "Linux Unified Key Setup (LUKS) verwenden". Nur SAN	„“
`tieringPolicy`	Tiering-Richtlinie zur Verwendung von „keiner“	„Nur Snapshot“ für eine ONTAP 9.5 SVM-DR-Konfiguration
`unixPermissions`	Modus für neue Volumes. **Leere leer für SMB Volumen.**	„“
`securityStyle`	Sicherheitstyp für neue Volumes. NFS unterstützt `mixed` Und `unix` Sicherheitsstile. SMB unterstützt `mixed` Und `ntfs` Sicherheitsstile.	NFS-Standard ist `unix`. SMB-Standard 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 Backend. Der `TridentBackendConfig` Mit Custom Resource Definition (CRD) können Sie Trident Back-Ends direkt über die Kubernetes Schnittstelle erstellen und managen. Dies können Sie mit tun `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 Löschrichtlinie 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: password
```

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)	Endpunkt	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

Beschreibung der geheimen Felder der Speicherplattform	Geheim	Feldbeschreibung
ONTAP	KundenPrivateKey	Base64-kodierte Wert des privaten Client-Schlüssels. Wird für die zertifikatbasierte Authentifizierung verwendet
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 verwiesen `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                     BACKEND NAME          BACKEND UUID
PHASE   STATUS
backend-tbc-ontap-san   ontap-san-backend     8d24fce7-6f60-4d4a-8ef6-
bab2699e6ab8   Bound   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's 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's `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 eine Intervention des Benutzers 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                    BACKEND NAME        BACKEND UUID
PHASE   STATUS    STORAGE DRIVER   DELETION POLICY
backend-tbc-ontap-san   ontap-san-backend   8d24fce7-6f60-4d4a-8ef6-
bab2699e6ab8   Bound   Success   ontap-san        delete
```

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".

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. In diesem Fall `TridentBackendConfig` Objekte können direkt über Kubernetes aktualisiert werden.

```
kubectl apply -f <updated-backend-file.yaml>
```

Alternativ können Sie Änderungen an der vorhandenen vornehmen `TridentBackendConfig` CR durch Ausführen des folgenden Befehls:

```
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 ausführen `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. 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 kein A 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 `kubectl` Und nicht `tridentctl`.

Um ein bereits vorhandenes Backend mit zu verwalten `kubectl`, 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 -n trident 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 [die UUID des Backend beachten].

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.

Management von Storage-Klassen

Suchen Sie Informationen zum Erstellen einer Speicherklasse, Löschen einer Speicherklasse und Anzeigen vorhandener Speicherklassen.

Entwurf einer Storage-Klasse

Siehe "Hier" Finden Sie weitere Informationen darüber, welche Storage-Klassen es gibt und wie Sie sie konfigurieren.

Erstellen Sie eine Speicherklasse

Führen Sie den folgenden Befehl aus, wenn Sie eine Speicherklassendatei haben:

```
kubectl create -f <storage-class-file>
```

`<storage-class-file>` Sollte durch den Dateinamen der Speicherklasse ersetzt werden.

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.

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 Standardspeicherklasse 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.

Sie sollten immer nur eine Standard-Storage-Klasse in Ihrem Cluster verwenden. 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}]'
```

Durchführung von Volume-Vorgängen

Erfahren Sie mehr über die Funktionen von Astra Trident zum Management Ihrer Volumes.

- "Verwenden Sie die CSI-Topologie"
- "Arbeiten Sie mit Snapshots"
- "Erweitern Sie Volumes"
- "Volumes importieren"

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". 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 dass sie von den Planungsanforderungen des anfragenden Pods abhängig sind.
- 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","kube
rnetes.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","kube
rnetes.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","kube
rnetes.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-central1
managementLIF: 172.16.238.5
svm: nfs_svm
username: admin
password: password
supportedTopologies:
- topology.kubernetes.io/region: us-central1
  topology.kubernetes.io/zone: us-central1-a
- topology.kubernetes.io/region: us-central1
  topology.kubernetes.io/zone: us-central1-b
storage:
- labels:
    workload: production
  region: Iowa-DC
  zone: Iowa-DC-A
  supportedTopologies:
  - topology.kubernetes.io/region: us-central1
    topology.kubernetes.io/zone: us-central1-a
- labels:
    workload: dev
  region: Iowa-DC
  zone: Iowa-DC-B
  supportedTopologies:
  - topology.kubernetes.io/region: us-central1
    topology.kubernetes.io/zone: us-central1-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:

```
kubectl create -f pvc.yaml
persistentvolumeclaim/pvc-san created
kubectl get pvc
NAME      STATUS    VOLUME   CAPACITY   ACCESS MODES   STORAGECLASS
AGE
pvc-san   Pending                                      netapp-san-us-east1
2s
kubectl 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                         Message
  ----    ------                ----  ----                         -------
  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: vol1
    persistentVolumeClaim:
      claimName: pvc-san
  containers:
  - name: sec-ctx-demo
    image: busybox
    command: [ "sh", "-c", "sleep 1h" ]
    volumeMounts:
    - name: vol1
      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 VolumeSnapshots (Volume Snapshot) von persistenten Volumes (PVS) können erstellt werden, um zeitpunktgenaue Kopien von Astra Trident Volumes zu erhalten. Darüber hinaus können Sie aus einem vorhandenen Volume-Snapshot ein neues Volume, auch als „*Clone*“ bekannt, erstellen. 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 müssen über einen externen Snapshot-Controller und benutzerdefinierte Ressourcen-Definitionen (CRDs) verfügen. 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 Bereitstellung eines Volume Snapshot Controllers.

> Erstellen Sie keinen Snapshot-Controller, wenn Sie On-Demand-Volumen-Schnappschüsse in einer GKE-Umgebung erstellen. GKE verwendet einen integrierten, versteckten Snapshot-Controller.

Schritt 1: Erstellen Sie ein `VolumeSnapshotClass`

Dieses Beispiel erstellt eine Volume-Snapshot-Klasse.

```
cat snap-sc.yaml
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshotClass
metadata:
  name: csi-snapclass
driver: csi.trident.netapp.io
deletionPolicy: Delete
```

Der `driver` Zeigt auf den CSI-Treiber von Astra Trident. `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.

Weitere Informationen finden Sie unter Link:../trident-reference/Objects.HTML#kubernetes-volumesnapshotclass-Objects[`VolumeSnapshotClass`].

Schritt 2: Erstellen Sie einen Schnappschuss eines vorhandenen PVC

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 der Snapshot für ein PVC mit dem Namen erstellt `pvc1` Der Name des Snapshots lautet `pvc1-snap`.

```
kubectl create -f snap.yaml
volumesnapshot.snapshot.storage.k8s.io/pvc1-snap created

kubectl get volumesnapshots
NAME                   AGE
pvc1-snap              50s
```

Dadurch wurde ein erstellt `VolumeSnapshot` Objekt: Ein VolumeSnapshot ist analog zu einem PVC und einem zugeordnet `VolumeSnapshotContent` Objekt, das den tatsächlichen Snapshot darstellt.

Es ist möglich, die zu identifizieren `VolumeSnapshotContent` Objekt für das `pvc1-snap` VolumeSnapshot wird beschrieben.

```
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
.
.
```

Der `Snapshot Content Name` Identifiziert das VolumeSnapshotContent-Objekt, das diesen Snapshot bereitstellt. Der `Ready To Use` Der Parameter gibt an, dass der Snapshot zum Erstellen einer neuen PVC verwendet werden kann.

Schritt 3: PVCs aus VolumeSnapshots erstellen

Das Beispiel erstellt ein PVC anhand eines Snapshots:

```
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: pvc1-snap
    kind: VolumeSnapshot
    apiGroup: snapshot.storage.k8s.io
```

`dataSource` Zeigt an, dass das PVC mit dem Namen VolumeSnapshot erstellt werden muss `pvc1-snap` Als Quelle der Daten. Damit beauftragt Astra Trident, aus dem Snapshot ein PVC zu erstellen. Nachdem die PVC erstellt wurde, kann sie an einem Pod befestigt und wie jedes andere PVC verwendet werden.

> Wenn Sie ein persistentes Volume mit zugeordneten Snapshots löschen, wird das entsprechende Trident-Volume in einen „Löschzustand“ aktualisiert. Damit das Astra Trident Volume gelöscht werden kann, sollten die Snapshots des Volume entfernt werden.

Bereitstellung eines Volume Snapshot Controllers

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 im gewünschten Namespace. Bearbeiten Sie die YAML-Manifeste unten, um den Namespace zu ändern.

```
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
```

Weiterführende Links

- "Volume Snapshots"
- "VolumeSnapshotKlasse"

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.

Überblick

Die Erweiterung eines iSCSI-PV umfasst die folgenden Schritte:

- Bearbeiten der StorageClass-Definition zum Festlegen des `allowVolumeExpansion` Feld an `true`.

- Bearbeiten der PVC-Definition und Aktualisieren der `spec.resources.requests.storage` Um die neu gewünschte Größe zu reflektieren, die größer als die ursprüngliche Größe sein muss.
- Das Anbringen des PV muss an einen Pod angehängt werden, damit die Größe geändert werden kann. 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-Back-End, 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.

Das folgende Beispiel zeigt, wie die Erweiterung von iSCSI PVS funktioniert.

Schritt: Storage Class für Volume-Erweiterung konfigurieren

```
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

```
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).

```
kubectl get pvc
NAME      STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS   AGE
san-pvc   Bound    pvc-8a814d62-bd58-4253-b0d1-82f2885db671   1Gi
RWO            ontap-san      8s

kubectl 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

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:

```
kubectl 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
kubectl 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-Volume

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 bearbeiten `kubectl edit storageclass` 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:

```
kubectl 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: Validieren Sie die Erweiterung

Sie können die korrekte Größenänderung validieren, indem Sie die Größe des PVC, des PV und des Astra Trident Volume überprüfen:

```
kubectl get pvc ontapnas20mb
NAME           STATUS   VOLUME
CAPACITY   ACCESS MODES   STORAGECLASS    AGE
ontapnas20mb   Bound    pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7   1Gi
RWO            ontapnas        4m44s

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   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`.

Treiber, die den Volumenimport unterstützen

In dieser Tabelle sind die Treiber aufgeführt, die den Import von Volumes unterstützen, und die Version, in der sie eingeführt wurden.

Treiber	Freigabe
`ontap-nas`	19.04
`ontap-nas-flexgroup`	19.04
`solidfire-san`	19.04
`azure-netapp-files`	19.04

Treiber	Freigabe
gcp-cvs	19.04
ontap-san	19.04

Warum sollte ich Volumes importieren?

Es gibt verschiedene Anwendungsfälle für den Import eines Volumes in Trident:

- Eine Anwendung erreichen und ihren vorhandenen Datensatz erneut verwenden
- Verwenden eines Klons eines Datensatzes für eine kurzlebige Anwendung
- Neuerstellung eines fehlerhaften Kubernetes-Clusters
- Migration von Applikationsdaten während der Disaster Recovery

Wie funktioniert der Import?

Die PVC-Datei (Persistent Volume Claim) wird vom Importprozess des Volumes zur Erstellung des PVC verwendet. Die PVC-Datei sollte mindestens die Felder Name, Namespace, accessModes und storageClassName enthalten, wie im folgenden Beispiel dargestellt.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: my_claim
  namespace: my_namespace
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: my_storage_class
```

Der `tridentctl` Der Client wird verwendet, um ein vorhandenes Storage Volume zu importieren. Trident importiert das Volume, indem Volume-Metadaten gespeichert und die PVC und das PV erstellt werden.

```
tridentctl import volume <backendName> <volumeName> -f <path-to-pvc-file>
```

Zum Importieren eines Storage-Volumes geben Sie den Namen des Astra Trident Backends mit dem Volume sowie den Namen an, der das Volume auf dem Storage eindeutig identifiziert (z. B. ONTAP FlexVol, Element Volume, CVS Volume Path). Das Storage-Volume muss Lese-/Schreibzugriff ermöglichen und über das angegebene Astra Trident-Back-End zugänglich sein. Der `-f` String Argument ist erforderlich und gibt den Pfad zur YAML- oder JSON-PVC-Datei an.

Erhält Astra Trident die Anfrage für das Importvolumen, wird die vorhandene Volume-Größe festgelegt und im PVC 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.

Wenn ein Volume mit dem importiert wird `--no-manage` Argument: Trident führt für den Lebenszyklus der Objekte keine zusätzlichen Operationen an der PVC oder PV durch. Da Trident PV- und PVC-Ereignisse für ignoriert `--no-manage` Objekte, das Speichervolumen wird nicht gelöscht, wenn das PV gelöscht wird. Andere Vorgänge, wie z. B. der Volume-Klon und die Volume-Größe, werden ebenfalls ignoriert. 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.

Trident 19.07 und höher verarbeiten den Anhang von PVS und mountet das Volume im Rahmen des Imports. Bei Importen mit früheren Versionen von Astra Trident gibt es keine Vorgänge im Datenpfad. Der Volume-Import überprüft nicht, ob das Volume gemountet werden kann. Wenn beim Import des Volumes ein Fehler gemacht wird (beispielsweise ist StorageClass falsch), können Sie die Zurückgewinnungsrichtlinie für das PV in wiederherstellen `retain`, Löschen der PVC und PV, und Wiederversuchen des Volumenimportbefehls.

`ontap-nas` Und `ontap-nas-flexgroup` Importe

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.

> Ein ONTAP Volume muss vom Typ rw aufweisen, um von Trident zu importieren. Wenn ein Volume vom Typ dp verwendet wird, es ein SnapMirror Ziel-Volume ist. Sie sollten die gespiegelte Beziehung unterbrechen, bevor Sie das Volume in Trident importieren.

> Der `ontap-nas` Der Treiber kann qtrees nicht importieren und verwalten. Der `ontap-nas` Und `ontap-nas-flexgroup` Treiber erlauben keine doppelten Volume-Namen.

Zum Beispiel, um ein Volume mit dem Namen zu importieren `managed_volume` Auf einem Backend mit dem Namen `ontap_nas`, Verwenden Sie den folgenden Befehl:

```
tridentctl import volume ontap_nas managed_volume -f <path-to-pvc-file>

+------------------------------------------+---------+---------------
+----------+------------------------------------------+--------+---------+
|                   NAME                   |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+------------------------------------------+---------+---------------
+----------+------------------------------------------+--------+---------+
| pvc-bf5ad463-afbb-11e9-8d9f-5254004dfdb7 | 1.0 GiB | standard      |
file     | c5a6f6a4-b052-423b-80d4-8fb491a14a22 | online | true    |
+------------------------------------------+---------+---------------
+----------+------------------------------------------+--------+---------+
```

So importieren Sie ein Volume mit dem Namen `unmanaged_volume` (Auf dem `ontap_nas backend`), die Trident nicht verwaltet, verwenden Sie den folgenden Befehl:

```
tridentctl import volume nas_blog unmanaged_volume -f <path-to-pvc-file>
--no-manage

+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
|                   NAME                  |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID              | STATE  | MANAGED |
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
| pvc-df07d542-afbc-11e9-8d9f-5254004dfdb7 | 1.0 GiB | standard      |
file     | c5a6f6a4-b052-423b-80d4-8fb491a14a22 | online | false   |
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
```

Bei Verwendung des `--no-manage` Argument: Trident umbenannt oder validiert nicht, ob das Volume angehängt war. Der Volumenimport schlägt fehl, wenn das Volume nicht manuell gemountet wurde.

Ein zuvor vorhandener Fehler beim Importieren von Volumes mit benutzerdefinierten UnixPermissions wurde behoben. Sie können unixPermissions in Ihrer PVC-Definition oder Back-End-Konfiguration angeben und Astra Trident anweisen, das Volume entsprechend zu importieren.

`ontap-san` Importieren

Astra Trident kann auch ONTAP SAN FlexVols importieren, die eine einzelne LUN enthalten. Dies entspricht dem `ontap-san` Treiber, der für jede PVC und eine LUN innerhalb der FlexVol eine FlexVol erstellt. Sie können das verwenden `tridentctl import` Befehl in gleicher Weise wie in anderen Fällen:

- Geben Sie den Namen des an `ontap-san` Back-End:
- Geben Sie den Namen der zu importierenden FlexVol an. Beachten Sie, dass diese FlexVol nur eine LUN enthält, die importiert werden muss.
- Geben Sie den Pfad der PVC-Definition an, die mit dem verwendet werden muss `-f` Flagge.
- Wählen Sie zwischen PVC-Verwaltung oder -Management. Standardmäßig verwaltet Trident die PVC und benennt die FlexVol und LUN auf dem Back-End um. Um als nicht verwaltetes Volume zu importieren, übergeben Sie den `--no-manage` Flagge.

Beim Importieren eines nicht verwalteten `ontap-san` Volume, Sie sollten sicherstellen, dass die LUN in der FlexVol benannt ist `lun0` Und ist einer Initiatorgruppe mit den gewünschten Initiatoren zugeordnet. Astra Trident übernimmt dies automatisch für einen verwalteten Import.

Astra Trident importiert dann den FlexVol und verknüpft ihn mit der PVC-Definition. Astra Trident ist auch für die FlexVol bekannt `pvc-<uuid>` Formatieren Sie und die LUN innerhalb der FlexVol bis `lun0`.

Es wird empfohlen, Volumes zu importieren, die keine aktiven Verbindungen haben. Wenn Sie ein aktiv verwendetes Volume importieren möchten, klonen Sie zuerst das Volume und führen Sie dann den Import durch.

Beispiel

Um den zu importieren `ontap-san-managed` FlexVol, die auf dem vorhanden ist `ontap_san_default` Back-End, führen Sie das aus `tridentctl import` Befehl als:

```
tridentctl import volume ontapsan_san_default ontap-san-managed -f pvc-
basic-import.yaml -n trident -d

+-----------------------------------------+--------+---------------
+----------+-----------------------------------------+--------+---------+
|                   NAME                  |  SIZE  | STORAGE CLASS |
PROTOCOL |             BACKEND UUID              | STATE  | MANAGED |
+-----------------------------------------+--------+---------------
+----------+-----------------------------------------+--------+---------+
| pvc-d6ee4f54-4e40-4454-92fd-d00fc228d74a | 20 MiB | basic         |
block    | cd394786-ddd5-4470-adc3-10c5ce4ca757 | online | true    |
+-----------------------------------------+--------+---------------
+----------+-----------------------------------------+--------+---------+
```

Ein ONTAP-Volume muss vom Typ rw sein, um von Astra Trident importiert werden zu können. Wenn ein Volume vom Typ dp ist, ist es ein SnapMirror Ziel-Volume. Sie sollten die Spiegelbeziehung brechen, bevor Sie das Volume in Astra Trident importieren.

`element` Importieren

Mit Trident können Sie NetApp Element Software/NetApp HCI Volumes in Ihr Kubernetes Cluster importieren. Sie brauchen den Namen Ihres Astra Trident Backend, und den eindeutigen Namen des Volumes und der PVC-Datei als Argumente für die `tridentctl import` Befehl.

```
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    |
+-----------------------------------------+--------+--------------
+----------+-----------------------------------------+--------+---------+
```

Der Elementtreiber unterstützt doppelte Volume-Namen. Wenn es doppelte Volume-Namen gibt, gibt Trident Volume Import Prozess einen Fehler zurück. Als Workaround können Sie das Volume klonen und einen eindeutigen Volume-Namen bereitstellen. Importieren Sie dann das geklonte Volume.

`gcp-cvs` Importieren

Für den Import eines durch die NetApp Cloud Volumes Service in GCP gesicherten Volumes sollten Sie das Volume nach seinem Volume-Pfad anstelle seines Namens identifizieren.

Um einen zu importieren `gcp-cvs` Datenträger auf dem Back-End aufgerufen `gcpcvs_YEppr` Mit dem Volume-Pfad von `adroit-jolly-swift`, Verwenden Sie den folgenden Befehl:

```
tridentctl import volume gcpcvs_YEppr adroit-jolly-swift -f <path-to-pvc-
file> -n trident

+-----------------------------------------+--------+--------------
+----------+-----------------------------------------+--------+---------+
|                   NAME                  |  SIZE  | STORAGE CLASS |
PROTOCOL |             BACKEND UUID              | STATE  | MANAGED |
+-----------------------------------------+--------+--------------
+----------+-----------------------------------------+--------+---------+
| pvc-a46ccab7-44aa-4433-94b1-e47fc8c0fa55 | 93 GiB | gcp-storage   | file
| e1a6e65b-299e-4568-ad05-4f0a105c888f | online | true    |
+-----------------------------------------+--------+--------------
+----------+-----------------------------------------+--------+---------+
```

Der Volume-Pfad ist der Teil des Exportpfads des Volumes nach dem :/. Beispiel: Wenn der Exportpfad lautet `10.0.0.1:/adroit-jolly-swift`, Der Volume-Pfad ist `adroit-jolly-swift`.

`azure-netapp-files` Importieren

Um einen zu importieren `azure-netapp-files` Datenträger auf dem Back-End aufgerufen `azurenetappfiles_40517` Mit dem Volume-Pfad `importvol1`, Ausführen des folgenden Befehls:

```
tridentctl import volume azurenetappfiles_40517 importvol1 -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    |
+----------------------------------------+---------+--------------
+----------+----------------------------------------+--------+---------+
```

Der Volume-Pfad für das ANF-Volumen ist im Mount-Pfad nach dem :/ vorhanden. Beispiel: Wenn der Mount-Pfad lautet `10.0.0.2:/importvol1`, Der Volume-Pfad ist `importvol1`.

Ein NFS-Volume kann über Namespaces hinweg genutzt werden

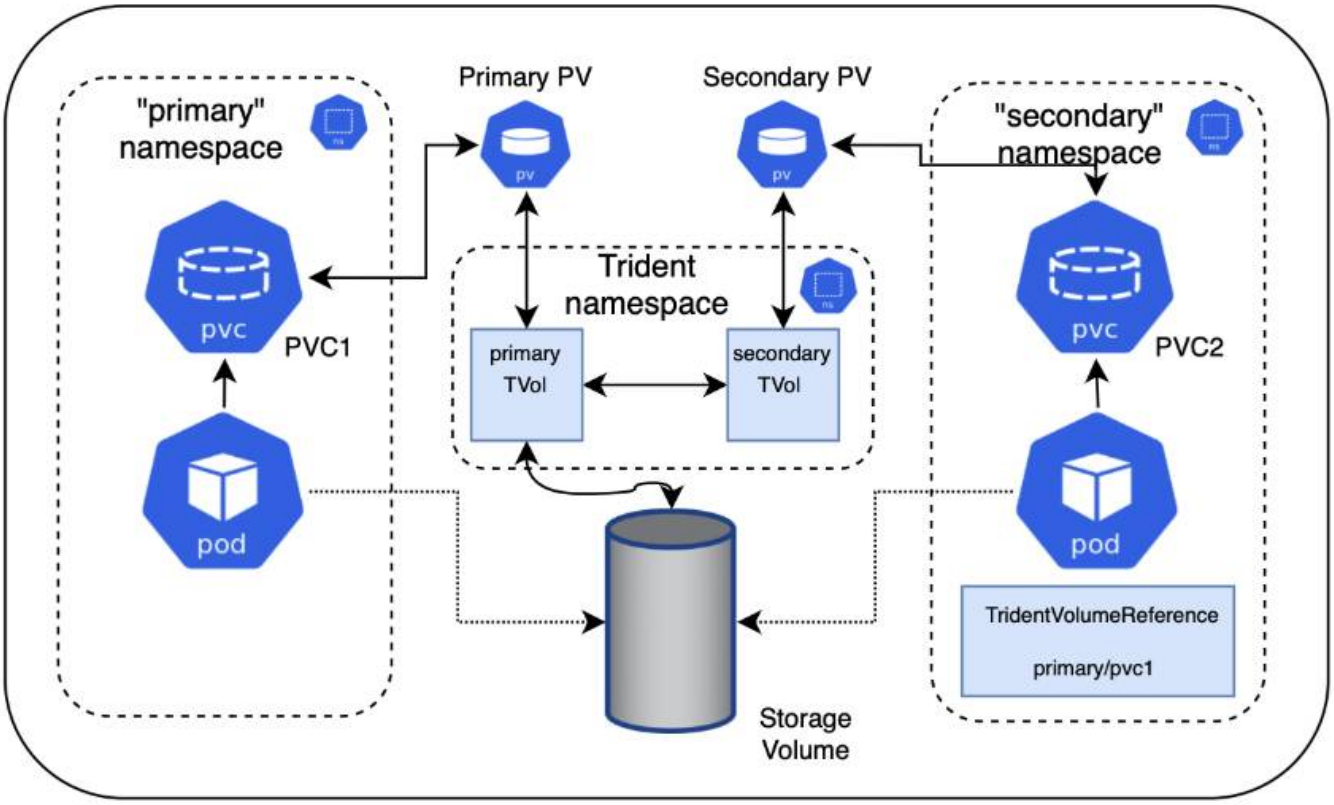
Mit Astra Trident können Sie ein Volume in einem primären Namespace erstellen und es in einem oder mehreren sekundären Namespaces teilen.

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[`tridentctl` Befehle und Optionen].

```
Usage:
  tridentctl get [option]
```

Markierungen:

- `-h, --help`: Hilfe für Volumen.
- `--parentOfSubordinate string`: Abfrage auf untergeordnetes Quellvolumen begrenzen.
- `--subordinateOf string`: Abfrage auf Untergebene beschränken.

Einschränkungen

- Astra Trident kann nicht verhindern, dass Ziel-Namespaces 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".
- Sehen Sie sich die Demo an "NetAppTV".

Überwachen Sie Astra Trident

Astra Trident bietet eine Reihe von Prometheus-Kennzahlendpunkten, mit denen Sie die Leistung von Astra Trident überwachen können.

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. Der Blog erläutert, wie Sie Prometheus als Operator in Ihrem Kubernetes Cluster und die Erstellung eines ServiceMonitor ausführen können, um die Kennzahlen von Astra Trident 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` endpunkt des Dienstes: Das Ergebnis: Prometheus ist jetzt so konfiguriert, dass sie die Kennzahlen von Astra Trident verstehen.

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 akzeptierenhttps://www.netapp.com/company/legal/privacy-policy/["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, der zusammen mit Astra Trident installiert wird Sie können das Container-Image unter abrufen "Trident AutoSupport".
- Trident AutoSupport erfasst oder übermittelt keine personenbezogenen Daten oder personenbezogenen Daten. Sie wird mit einem geliefert "EULA" , das sich nicht für das Trident Container-Image selbst eignet. Weitere Informationen zum Engagement von NetApp für Datensicherheit und Vertrauen finden "Hier"Sie 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 erfolgen `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:

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	`sudo yum install -y nfs-utils`
NFS	Ubuntu	`sudo apt-get install -y nfs-common`

Protokoll	Betriebssystem	Befehle

ISCSI	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`

Protokoll	Betriebssystem	Befehle

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`

Implementieren Sie Astra Trident

```
sudo systemctl enable
--now open-
iscsi.service
sudo systemctl enable
open-iscsi
```

Astra Trident for Docker bietet eine direkte Integration in das Docker Ecosystem für die Storage-Plattformen von NetApp. 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:

```
pkill /usr/local/bin/netappdvp
pkill /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 angeben `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/v23.01.1/trident-
      installer-23.01.1.tar.gz
      tar zxf trident-installer-23.01.1.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 bzw. die Konfigurationsdateien erstellt haben, starten Sie den Trident-Daemon mit der gewünschten Konfigurationsdatei.

```
sudo trident --config=/etc/netappdvp/ontap-nas.json
```

Sofern nicht angegeben, ist 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 an `-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
`size`	Die Größe des Volumes beträgt standardmäßig 1 gib.
`spaceReserve`	Thin oder Thick Provisioning stellen das Volume bereit. Die Standardeinstellung ist „Thin“. Gültige Werte sind `none` (Thin Provisioning) und `volume` (Thick Provisioning):
`snapshotPolicy`	Dadurch wird die Snapshot-Richtlinie auf den gewünschten Wert eingestellt. Die Standardeinstellung lautet `none`, 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 `.snapshot` 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 `false`. 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. Zum Beispiel kann das Klonen einer leeren Datenbank große Zeitersparnis bieten, aber nur geringe Storage-Einsparungen. Daher ist es am besten, den Klon sofort zu teilen.
`encryption`	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. 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".
`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
`unixPermissions`	Dadurch wird der Berechtigungssatz für das Volume selbst festgelegt. Standardmäßig werden die Berechtigungen auf festgelegt `---rwxr-xr-x`, Oder in numerischer Notation 0755, und `root` Wird der Eigentümer sein. Das Text- oder Zahlenformat funktioniert.
`snapshotDir`	Einstellen auf `true` Wird die schaffen `.snapshot` Für Clients, die auf das Volume zugreifen, sichtbares Verzeichnis Der Standardwert ist `false`, Das bedeutet, dass die Sichtbarkeit des `.snapshot` Das Verzeichnis ist standardmäßig deaktiviert. Einige Bilder, zum Beispiel das offizielle MySQL-Bild, funktionieren nicht wie erwartet, wenn die `.snapshot` Verzeichnis wird angezeigt.
`exportPolicy`	Legt die Exportrichtlinie fest, die für das Volume verwendet werden soll. Die Standardeinstellung lautet `default`.
`securityStyle`	Legt den Sicherheitsstil für den Zugriff auf das Volume fest. Die Standardeinstellung lautet `unix`. Gültige Werte sind `unix` Und `mixed`.

Die folgenden zusätzlichen Optionen sind für iSCSI **nur**:

Option	Beschreibung
`fileSystemType`	Legt das Dateisystem fest, das zum Formatieren von iSCSI-Volumes verwendet wird. Die Standardeinstellung lautet `ext4`. Gültige Werte sind `ext3`, `ext4`, und `xfs`.
`spaceAllocation`	Einstellen auf `false` Wird die Raumzuordnungsfunktion der LUN deaktivieren. Der Standardwert ist `true`, 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 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 in 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 Instanziieren 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 Volume-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. Der `size` Die Option 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`	IP-Adresse des LIF-Protokolls. **ONTAP NAS drivers**: Wir empfehlen die Angabe `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. **ONTAP-SAN-Treiber**: Geben Sie nicht für iSCSI an. Astra Trident verwendet "ONTAP selektive LUN-Zuordnung" Um die iSCI LIFs zu ermitteln, die für die Einrichtung einer Multi-Path-Sitzung erforderlich sind. Wenn eine Warnung erzeugt wird `dataLIF` Ist explizit definiert.	`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 `ontap-nas-flexgroup` 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`	Die vom Plug-in verwendete Initiatorgruppe wird standardmäßig verwendet `netappdvp`. **Nur erhältlich für den `ontap-san`dFluss**.	`myigroup`
`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`
`snapshotReserve`	Der Prozentsatz der Snapshot-Reserve ist standardmäßig „“, um den ONTAP-Standardwert zu akzeptieren	`10`

Option	Beschreibung	Beispiel
`splitOnClone`	Teilen Sie einen Klon bei der Erstellung von seinem übergeordneten Element auf. Dies ist standardmäßig der Standardwert `false`	`false`
`encryption`	Aktiviert 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. 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".	Richtig
`unixPermissions`	NAS-Option für bereitgestellte NFS-Volumes, standardmäßig auf `777`	`777`
`snapshotDir`	NAS-Option für den Zugriff auf die `.snapshot` Verzeichnis, Standardeinstellung `false`	`true`
`exportPolicy`	NAS-Option für die zu verwendende NFS-Exportrichtlinie, standardmäßig auf `default`	`default`
`securityStyle`	NAS-Option für Zugriff auf das bereitgestellte NFS-Volume. NFS unterstützt `mixed` Und `unix` Sicherheitsstile. Die Standardeinstellung lautet `unix`.	`unix`
`fileSystemType`	SAN-Option zum Auswählen des Dateisystemtyps, standardmäßig auf `ext4`	`xfs`

Option	Beschreibung	Beispiel
`tieringPolicy`	Zu verwendende Tiering-Richtlinie, Standard ist `none`; `snapshot-only` Für Konfiguration vor ONTAP 9.5 SVM-DR	`none`

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": "myigroup"
}
```

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": "myigroup"
}
```

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>@<mvip>/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 haben und ein Upgrade mit vorhandenen Volumes durchführen, müssen Sie diesen Wert festlegen, um auf Ihre 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.

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. Trident 20.01 führt Prometheus Endpunkte ein, mit denen Informationen über den Betrieb von Astra Trident erfasst werden können. Dazu zählen die Anzahl der verwalteten Back-Ends, die Anzahl der bereitgestellten Volumes, die verbrauchten Bytes usw. Außerdem können Sie Cloud Insights zur Überwachung und Analyse einsetzen.

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

Welche Versionen werden von unterstützt `etcd`?

Astra Trident muss nicht mehr ein `etcd`. Er verwendet CRDs, um den Status beizubehalten.

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 Replikate 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 das PersistentVolume, PersistVolumeClaim, StorageClass und Secret Ressourcen 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 es wiederherstellen?

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. Die 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?

Es gibt eine Reihe von Faktoren, die bewertet werden müssen, wenn Sie herunterstufen möchten. Siehe "Der Abschnitt zum Downgrade".

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 iSCI 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. Ab 20.04 unterstützt Astra Trident bidirektionale CHAP-Back-Ends für ONTAP. 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. Diese Funktion erfordert CSI Trident.

Können wir einen Port im DataLIF angeben?

Astra Trident 19.01 und höher unterstützt die Angabe eines Ports in der DataLIF. Konfigurieren Sie es im `backend.json` Datei als `"managementLIF": <ip address>:<port>"`. Wenn die IP-Adresse Ihres Management LIF beispielsweise 192.0.2.1 ist und der Port 1000 ist, konfigurieren Sie `"managementLIF": "192.0.2.1:1000"`.

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 installiert sein `--use-ipv6` Flag für Funktion über IPv6.

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 es in Astra Trident importieren, wenn ich ein Volume habe, das außerhalb von Astra Trident erstellt wurde?

Ab Version 19.04 können Sie Volumes mit der Importfunktion für Volumes in Kubernetes einbringen.

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.
```

Kann ich die von Astra Trident erstellten iSCSI PVCs erweitern?

Trident 19.10 unterstützt die Erweiterung von iSCSI PVS mithilfe von CSI-Bereitstellung.

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 Volumen 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.

Unterstützung

Astra Trident ist ein offiziell unterstütztes NetApp Projekt. Sie können sich mit einem der Standardmechanismen an NetApp wenden und erhalten den benötigten Support der Enterprise-Klasse.

Auf unserem Astra ist auch eine lebendige Public Community von Container-Nutzern (einschließlich Astra Trident Entwickler) verfügbar "Kanal abstecken". Hier können Sie allgemeine Fragen zum Projekt stellen und verwandte Themen mit Gleichgesinnten diskutieren.

Fehlerbehebung

Verwenden Sie die hier angegebenen Hinweise zur Fehlerbehebung bei Problemen, die bei der Installation und Verwendung von Astra Trident möglicherweise auftreten können.

Astra Trident unterstützt Sie bei der Erstellung eines Support-Bundles mit `tridentctl logs -a -n trident` Senden an `NetApp Support <Getting Help>`.

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".

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 Sie ein Kubernetes-Cluster aktualisieren und/oder Trident verwenden, um Beta-Volume-Snapshots zu verwenden, stellen Sie sicher, dass alle vorhandenen Alpha-Snapshot-CRS vollständig entfernt werden. Anschließend können Sie die verwenden `tridentctl obliviate alpha-snapshot-crd` Befehl zum Löschen von Alpha-Snapshot-CRDs. Siehe "Diesem Blog" Die Schritte zu verstehen, die bei der Migration von Alpha-Snapshots erforderlich sind.
- 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 zunächst die aus `tridentctl uninstall` Befehl zum Entfernen von Trident. Laden Sie die gewünschten herunter "Trident Version" Und installieren Sie mit `tridentctl install` Befehl. Eine Herabstufung sollte nur in Betracht gezogen werden, wenn keine neuen PVS erstellt wurden und keine Änderungen an bereits vorhandenen PVS/Back-Ends/Storage-Klassen vorgenommen wurden. Da Trident jetzt CRDs für die Statushaltung verwendet, verfügen alle erstellten Storage-Einheiten (Back-Ends, Storage-Klassen, PVS und Volume Snapshots) über `associated CRD objects <Kubernetes CustomResourceDefinition Objects>` Anstatt Daten, die in das PV geschrieben wurden, die von der früheren installierten Version von Trident verwendet wurden. **Neu erstellte PVS können nicht verwendet werden, wenn sie auf eine frühere Version zurückverschoben werden. Änderungen an Objekten wie Back-Ends, PVS, Speicherklassen und Volume-Snapshots (erstellt/aktualisiert/gelöscht) werden Trident bei Downgraded** nicht sichtbar. Das PV, das von der früheren Version von Trident verwendet wurde, ist für Trident weiterhin sichtbar. Wenn Sie zu einer früheren Version zurückkehren, wird der Zugriff auf PVS, die bereits mit der älteren Version erstellt wurden, nicht unterbrochen, es sei denn, sie wurden aktualisiert.
- Um Trident vollständig zu entfernen, führen Sie den aus `tridentctl obliviate crd` Befehl. Dadurch werden alle CRD-Objekte entfernt und die CRDs werden nicht definiert. Trident verwaltet keine PVS mehr,

die bereits bereitgestellt wurden.

> Trident muss danach von Grund auf neu konfiguriert werden.

- 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, sollten Sie einen Blick auf das werfen `TridentOrchestrator` Status:

```
kubectl 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 über eine Instanz von Trident verfügen kann, stellt der Operator sicher, dass zu einem beliebigen Zeitpunkt nur eine aktive Instanz vorhanden ist `TridentOrchestrator` Die sie erstellen kann.

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
AGE
trident-csi-4p5kq                   1/2     ImagePullBackOff   0
5m18s
trident-csi-6f45bfd8b6-vfrkw        4/5     ImagePullBackOff   0
5m19s
trident-csi-9q5xc                   1/2     ImagePullBackOff   0
5m18s
trident-csi-9v95z                   1/2     ImagePullBackOff   0
5m18s
trident-operator-766f7b8658-ldzsv   1/1     Running            0
8m17s
```

Sie können klar sehen, dass die Pods nicht vollständig initialisiert werden können, da ein oder mehrere Container-Images 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 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 erneut ausführen `-d` 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 ausführen `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.
```

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. Trotzdem sollte die Management-LIF über DNS zugänglich sein, um maximale Flexibilität zu ermöglichen, sollte "SVM-DR" in Verbindung 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` Der Treiber erstellt ca. ein FlexVolum für alle 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. Bidirektionale CHAP-Unterstützung ist bei Trident 20.04 und höher verfügbar.

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 bei der "Verhindern Sie einen Schläger" Überwachung oder außer Kontrolle geraten durch Container, die Workloads außerhalb der Trident SVM beeinträchtigen.

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 Zielservern 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 Exportrichtlinien ab Version 20.04 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-Storage-Management" (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 Infrastrukturteams. 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	Snapshots	Klone	Dynamische Exportrichtlinien	Multi-Anschluss	QoS	Größe Ändern	Replizierung
`ontap-nas`	Ja.	Ja.	Yes [5]	Ja.	Yes [1]	Ja.	Yes [1]
`ontap-nas-economy`	Yes [3]	Yes [3]	Yes [5]	Ja.	Yes [3]	Ja.	Yes [3]
`ontap-nas-flexgroup`	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	Snapshots	Klone	Multi-Anschluss	Bidirektionales CHAP	QoS	Größe Ändern	Replizierung
`ontap-san`	Ja.	Ja.	Yes [4]	Ja.	Yes [1]	Ja.	Yes [1]
`ontap-san-economy`	Ja.	Ja.	Yes [4]	Ja.	Yes [3]	Ja.	Yes [3]

> Fußnote für die obigen Tabellen: Ja [1]: Not Managed by Astra Trident Ja [2]: Managed by Astra Trident, but not PV granular Ja [3]: Nicht verwaltet durch Astra Trident und nicht durch PV-Granularität Ja [4]: Unterstützt von RAW-Block-Volumes Ja [5]: Unterstützt von CSI 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 für ONTAP ermöglicht es Kunden, bereits bekannte NetApp Funktionen, Performance und Administration zu nutzen und gleichzeitig die Einfachheit, Agilität, Sicherheit und Skalierbarkeit beim Speichern von Daten in 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	Snapshots	Klone	Multi-Anschluss	CHAP	QoS	Größe Ändern	Replizierung
`solidfire-san`	Ja.	Ja.	Yes [2]	Ja.	Ja.	Ja.	Yes [1]

> Fußnote: Yes [1]: Nicht verwaltet durch Astra Trident Ja [2]: Wird für RAW-Block-Volumes unterstützt

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
`azure-netapp-files`	Ja.	Ja.	Ja.	Ja.	Ja.	Yes [1]

> Fußnote: Yes [1]: Nicht verwaltet durch Astra Trident

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
`gcp-cvs`	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 Auswahletiketten 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 ANF-Backend mit mehreren Etiketten, die unterschiedliche Leistungsstufen darstellen. 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. Jedoch, dies verhindert nicht, dass einige Aspekte des Volumens außerhalb von Kubernetes geändert werden. 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. Es fügt automatisch die konfigurierte PVC-Volumengröße hinzu (und überschreibt sie gegebenenfalls). 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 Bereitstellung von Protokollierung während der ersten OpenShift-Installation und die Bereitstellung von Protokollierung nach der Installation von OpenShift.

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 NFS-Server von ONTAP hat diese Probleme nicht und kann einfach eine Protokollierungs-Implementierung zurück. 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 an `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 entsprechend der Implementierungsmethode ist wichtig, um sicherzustellen, dass die zugrunde liegenden Services den richtigen Storage verwenden. Werfen wir einen Blick auf die Optionen für jede der Bereitstellungsmethoden.

Die nachfolgenden Tabellen enthalten nur die Variablen, die für die Storage-Konfiguration relevant sind, da sie 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
`openshift_logging_storage_kind`	Auf einstellen `nfs` So erstellen Sie ein NFS-PV für den Protokollierungsservice.
`openshift_logging_storage_host`	Der Hostname oder die IP-Adresse des NFS-Hosts. Diese Einstellung sollte auf die Daten-LIF für Ihre Virtual Machine eingestellt sein.
`openshift_logging_storage_nfs_directory`	Der Mount-Pfad für den NFS-Export. Beispiel: Wenn das Volume mit verbunden ist `/openshift_logging`, Sie würden diesen Pfad für diese Variable verwenden.
`openshift_logging_storage_volume_name`	Der Name, z.B. `pv_ose_logs`, Des zu erstellenden PV.
`openshift_logging_storage_volume_size`	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
`openshift_logging_es_pvc_dynamic`	Setzen Sie auf „true“, um dynamisch bereitgestellte Volumes zu verwenden.
`openshift_logging_es_pvc_storage_class_name`	Der Name der Speicherklasse, die in der PVC verwendet wird.
`openshift_logging_es_pvc_size`	Die Größe des im PVC angeforderten Volumens.
`openshift_logging_es_pvc_prefix`	Ein Präfix für die vom Protokollierungsservice verwendeten VES.
`openshift_logging_es_ops_pvc_dynamic`	Auf einstellen `true` Um dynamisch bereitgestellte Volumes für die OPS-Protokollierungsinstanz zu verwenden.
`openshift_logging_es_ops_pvc_storage_class_name`	Der Name der Speicherklasse für die OPS-Protokollierungsinstanz.
`openshift_logging_es_ops_pvc_size`	Die Größe der Volume-Anforderung für die OPS-Instanz.
`openshift_logging_es_ops_pvc_prefix`	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. Auch, wie der Protokollierungsservice, kann der Kennzahlendienst während einer Ersteinrichtung des Clusters oder nach seiner Inbetriebnahme mithilfe der Installationsmethode der 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
`openshift_metrics_storage_kind`	Auf einstellen `nfs` So erstellen Sie ein NFS-PV für den Protokollierungsservice.
`openshift_metrics_storage_host`	Der Hostname oder die IP-Adresse des NFS-Hosts. Diese Einstellung sollte auf die Daten-LIF für Ihre SVM eingestellt sein.
`openshift_metrics_storage_nfs_directory`	Der Mount-Pfad für den NFS-Export. Beispiel: Wenn das Volume mit verbunden ist `/openshift_metrics`, Sie würden diesen Pfad für diese Variable verwenden.
`openshift_metrics_storage_volume_name`	Der Name, z.B. `pv_ose_metrics`, Des zu erstellenden PV.
`openshift_metrics_storage_volume_size`	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
`openshift_metrics_cassandra_pvc_prefix`	Ein Präfix, das für die PVCs der Kennzahlen verwendet wird.
`openshift_metrics_cassandra_pvc_size`	Die Größe der Volumes, die angefordert werden sollen.
`openshift_metrics_cassandra_storage_type`	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.
`openshift_metrics_cassanda_pvc_storage_class_name`	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 nach der Installation von OpenShift mithilfe der Playbooks von Komponenten implementieren, werden in Ansible alle erforderlichen PVCs erstellt, und nachdem Astra Trident Storage für sie bereitgestellt hat, wird der Service implementiert.

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-Implementierungsleitfaden von Red hat" Für Ihre Version so konfigurieren, dass sie für Ihre Umgebung konfiguriert ist.

Datensicherung

Erfahren Sie mehr zu Datensicherungs- und Recovery-Optionen, die NetApp Storage-Plattformen bereitstellen. Astra Trident kann Volumes bereitstellen, die von einigen dieser Funktionen profitieren. Für jede Applikation mit einer Persistenzanforderung sollte eine Datensicherungs- und Recovery-Strategie eingesetzt werden.

Sichern Sie die `etcd` Cluster-Daten

Astra Trident speichert die Metadaten in den Kubernetes Clustern `etcd` Datenbank: Sichern Sie regelmäßig den `etcd` Cluster-Daten sind wichtig, um Kubernetes-Cluster unter Disaster-Szenarien wiederherzustellen.

Schritte

1. Der `etcdctl snapshot save` Mit dem Befehl können Sie einen zeitpunktgenauen Snapshot von erstellen `etcd` Cluster:

```
sudo docker run --rm -v /backup:/backup \
  --network host \
  -v /etc/kubernetes/pki/etcd:/etc/kubernetes/pki/etcd \
  --env ETCDCTL_API=3 \
  registry.k8s.io/etcd-amd64:3.2.18 \
  etcdctl --endpoints=https://127.0.0.1:2379 \
  --cacert=/etc/kubernetes/pki/etcd/ca.crt \
  --cert=/etc/kubernetes/pki/etcd/healthcheck-client.crt \
  --key=/etc/kubernetes/pki/etcd/healthcheck-client.key \
  snapshot save /backup/etcd-snapshot.db
```

Dieser Befehl erstellt einen etcd Snapshot, indem er einen etcd Container hochspend und speichert ihn im `/backup` Verzeichnis.

2. Bei einem Notfall können Sie einen Kubernetes-Cluster mithilfe der etcd Snapshots aufsetzen. Verwenden Sie die `etcdctl snapshot restore` Befehl zum Wiederherstellen eines bestimmten Snapshot, der in aufgenommen wurde `/var/lib/etcd` Ordner. Bestätigen Sie nach der Wiederherstellung, ob der `/var/lib/etcd` Der Ordner wurde mit dem ausgefüllt `member` Ordner. Im Folgenden finden Sie ein Beispiel für `etcdctl snapshot restore` Befehl:

```
etcdctl snapshot restore '/backup/etcd-snapshot-latest.db' ; mv
/default.etcd/member/ /var/lib/etcd/
```

3. Kopieren Sie vor dem Initialisieren des Kubernetes-Clusters alle erforderlichen Zertifikate.
4. Erstellen Sie den Cluster mit `--ignore-preflight-errors=DirAvailable—var-lib-etcd` Flagge.
5. Nachdem der Cluster aufleuchtet, stellen Sie sicher, dass die kube-System-Pods gestartet wurden.
6. Verwenden Sie die `kubectl get crd` Befehl zur Überprüfung, ob die von Trident erstellten benutzerdefinierten Ressourcen vorhanden sind und Trident Objekte abrufen, um sicherzustellen, dass alle Daten verfügbar sind.

Daten mit ONTAP Snapshots wiederherstellen

Snapshots spielen eine wichtige Rolle, indem sie zeitpunktgenaue Recovery-Optionen für Applikationsdaten bieten. Snapshots sind jedoch keine Backups allein. Sie schützen nicht vor Ausfällen oder anderen Katastrophen. Aber sie sind eine praktische, schnelle und einfache Möglichkeit, Daten in den meisten Szenarien wiederherzustellen. Erfahren Sie mehr darüber, wie Sie mit der ONTAP Snapshot Technologie Backups des Volumes erstellen und diese wiederherstellen können.

- Wenn die Snapshot-Richtlinie nicht im Backend definiert wurde, wird standardmäßig mit verwendet `none` Richtlinie: Dies führt dazu, dass ONTAP keine automatischen Snapshots erstellt. Der Storage-Administrator kann jedoch manuelle Snapshots erstellen oder die Snapshot-Richtlinien über die ONTAP Managementoberfläche ändern. Dies hat keine Auswirkung auf den Betrieb von Trident.
- Das Snapshot-Verzeichnis ist standardmäßig ausgeblendet. Dies erleichtert die maximale Kompatibilität von über den bereitgestellten Volumes `ontap-nas` Und `ontap-nas-economy` Treiber. Aktivieren Sie die `.snapshot` Verzeichnis bei Verwendung des `ontap-nas` Und `ontap-nas-economy` Treiber, mit denen Anwendungen Daten aus Snapshots direkt wiederherstellen können.

- Stellen Sie ein Volume in einen Zustand wieder her, der in einem vorherigen Snapshot mit dem aufgezeichnet wurde `volume snapshot restore` ONTAP-CLI-Befehl. Wenn Sie eine Snapshot Kopie wiederherstellen, überschreibt der Wiederherstellungsvorgang die vorhandene Volume-Konfiguration. Alle Änderungen an den Daten auf dem Volume nach der Erstellung der Snapshot Kopie gehen verloren.

```
cluster1::*> volume snapshot restore -vserver vs0 -volume vol3 -snapshot
vol3_snap_archive
```

Datenreplizierung mit ONTAP

Die Replizierung von Daten kann eine wichtige Rolle beim Schutz vor Datenverlusten aufgrund von Storage-Array-Ausfällen spielen.

Weitere Informationen zu den Replikationstechnologien von ONTAP finden Sie unter "ONTAP-Dokumentation" .

Replizierung von SnapMirror Storage Virtual Machines (SVM)

Sie können verwenden "SnapMirror", um eine vollständige SVM zu replizieren, einschließlich ihrer Konfigurationseinstellungen und ihrer Volumes. Bei einem Notfall können Sie die SnapMirror Ziel-SVM aktivieren, um die Datenbereitstellung zu starten. Sie können bei der Wiederherstellung der Systeme zurück auf das primäre System wechseln.

Astra Trident kann die Replizierungsbeziehungen nicht selbst konfigurieren, sodass der Storage-Administrator mithilfe der ONTAP SnapMirror SVM Replication-Funktion Volumes automatisch zu einem Disaster Recovery-Ziel (DR) replizieren kann.

Wenn Sie planen, die SnapMirror SVM Replication Funktion zu verwenden oder derzeit die Funktion nutzen, sollten Sie Folgendes berücksichtigen:

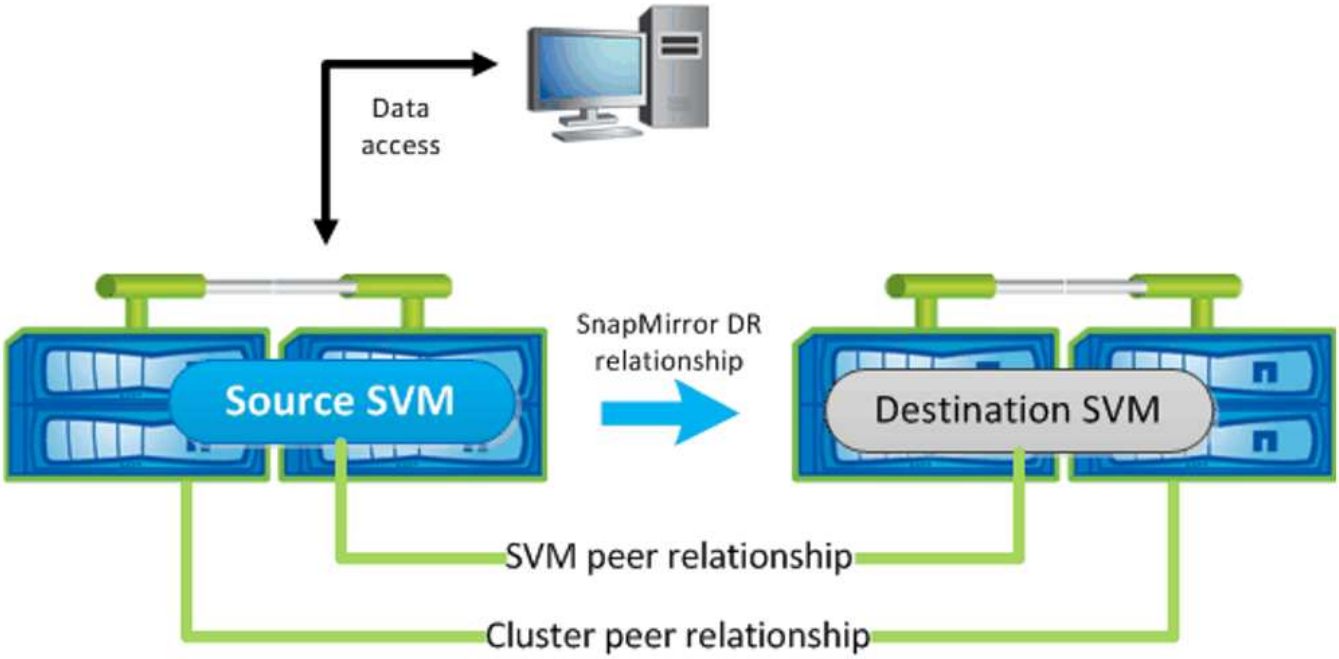
- Sie sollten ein gesondertes Back-End für jede SVM erstellen, bei der die SVM-DR aktiviert ist.
- Sie sollten die Speicherklassen so konfigurieren, dass Sie die replizierten Back-Ends nicht auswählen, außer wenn gewünscht. Dies ist wichtig, damit keine Volumes bereitgestellt werden müssen, die keine Sicherung einer Replizierungsbeziehung auf die Back-End(s) benötigen, die SVM-DR unterstützen.
- Applikationsadministratoren sollten die zusätzlichen Kosten und Komplexität verstehen, die mit der Replizierung der Daten verbunden sind. Außerdem sollte ein Recovery-Plan vor der Datenreplizierung ermittelt werden.
- Vor der Aktivierung der SnapMirror Ziel-SVM sollten alle geplanten SnapMirror-Transfers angehalten, alle laufenden SnapMirror Transfers abgebrochen, die Replizierungsbeziehung unterbrochen, die Quell-SVM beendet und dann die SnapMirror Ziel-SVM gestartet werden.
- Astra Trident erkennt SVM-Ausfälle nicht automatisch. Deshalb sollte der Administrator bei einem Ausfall das ausführen `tridentctl backend update` Befehl zum Auslösen von Trident-Failover auf das neue Backend.

Dies ist eine Übersicht der Schritte zur SVM-Einrichtung:

- Peering zwischen dem Quell- und Ziel-Cluster und der SVM einrichten.
- Erstellen Sie die Ziel-SVM mit `-subtype dp-destination` Option.
- Erstellen Sie einen Replikationsjob-Zeitplan, um sicherzustellen, dass die Replikation in den erforderlichen

Intervallen stattfindet.

- SnapMirror Replizierung von der Ziel-SVM zu der Quell-SVM mit der erstellen `-identity-preserve true` Option, um sicherzustellen, dass die SVM-Konfigurationen und Quell-SVM-Schnittstellen auf das Ziel kopiert werden. Initialisieren Sie die SnapMirror SVM-Replizierungsbeziehung von der Ziel-SVM.



Disaster Recovery Workflow für Trident

Astra Trident 19.07 und höher verwenden Kubernetes CRDs zum Speichern und Managen seines eigenen Status. Es verwendet den Kubernetes Cluster `etcd` Um die Metadaten zu speichern. Hier nehmen wir an, dass Kubernetes `etcd` Datendateien und die Zertifikate sind auf NetApp FlexVol gespeichert. Dieses FlexVol Volume befindet sich in einer SVM, die über eine SnapMirror SVM-DR-Beziehung mit einer Ziel-SVM am sekundären Standort verfügt.

Beschreiben Sie die folgenden Schritte, wie Sie bei einem Notfall ein einzelnes Kubernetes Cluster mit Astra Trident wiederherstellen können:

1. Wenn die Quell-SVM ausfällt, aktivieren Sie die SnapMirror Ziel-SVM. Dazu sollten Sie geplante SnapMirror Transfers anhalten, laufende SnapMirror Transfers abbrechen, die Replizierungsbeziehung unterbrechen, die Quell-SVM stoppen und die Ziel-SVM starten.
2. Mounten Sie das Volume, das den Kubernetes enthält, von der Ziel-SVM `etcd` Datendateien und Zertifikate auf dem Host, der als Master-Node eingerichtet wird.
3. Kopieren Sie alle erforderlichen Zertifikate zum Kubernetes-Cluster unter `/etc/kubernetes/pki` Und das usw. `member` Dateien unter `/var/lib/etcd`.
4. Erstellen Sie mit dem einen Kubernetes-Cluster `kubeadm init` Befehl mit dem `--ignore-preflight-errors=DirAvailable—var-lib-etcd` Flagge. Die für die Kubernetes-Nodes verwendeten Hostnamen sollten mit denen des Quell-Kubernetes-Clusters übereinstimmen.
5. Führen Sie die aus `kubectl get crd` Befehl zur Überprüfung, ob alle benutzerdefinierten Trident Ressourcen aufgerufen wurden, um zu überprüfen, ob alle Daten verfügbar sind.
6. Aktualisieren Sie alle erforderlichen Back-Ends, um den neuen Ziel-SVM-Namen wiederzugeben, indem

Sie das ausführen `./tridentctl update backend <backend-name> -f <backend-json-file> -n <namespace>` Befehl.

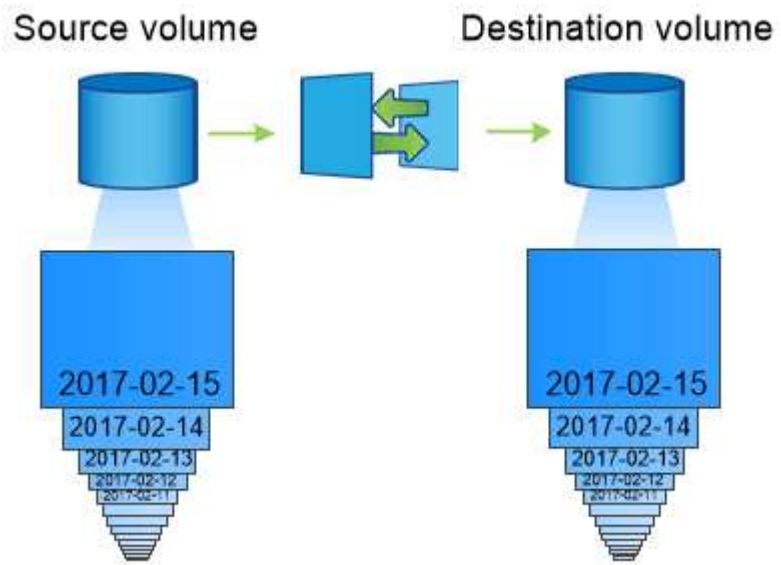
> Wenn die Ziel-SVM für persistente Applikations-Volumes aktiviert ist, stellen alle von Trident bereitgestellten Volumes Daten bereit. Nachdem der Kubernetes-Cluster mit den oben beschriebenen Schritten auf der Zielseite eingerichtet wurde, werden alle Implementierungen und Pods gestartet und die Container-Applikationen sollten ohne Probleme ausgeführt werden.

SnapMirror Volume-Replizierung

ONTAP SnapMirror Volume-Replizierung ist eine Disaster-Recovery-Funktion, die einen Failover auf Ziel-Storage von dem primären Storage auf Volume-Ebene ermöglicht. SnapMirror erstellt mithilfe der Synchronisierung von Snapshots ein Replikat oder eine Spiegelung des primären Storage für Volumes im sekundären Storage.

Dies ist ein Überblick über die Einrichtungsschritte für die ONTAP SnapMirror Volume-Replizierung:

- Peering zwischen den Clustern, in denen sich die Volumes befinden, und den SVMs, die Daten von den Volumes bereitstellen
- SnapMirror-Richtlinie erstellen, die das Verhalten der Beziehung steuert und die Konfigurationsattribute für diese Beziehung festlegt
- Erstellen Sie mit dem Befehl^] eine SnapMirror-Beziehung zwischen dem Ziel-Volume und dem Quell-Volume[`snapmirror create`, und weisen Sie die entsprechende SnapMirror-Richtlinie zu.
- Nach der Erstellung der SnapMirror Beziehung initialisieren Sie die Beziehung, damit ein Basistransfer vom Quell-Volume zum Ziel-Volume abgeschlossen wird.



SnapMirror Workflow für Disaster Recovery von Volumes für Trident

In den folgenden Schritten wird beschrieben, wie ein einzelner Kubernetes-Cluster mit Astra Trident wiederhergestellt wird.

1. Bei einem Ausfall alle geplanten SnapMirror-Transfers stoppen und alle laufenden SnapMirror Transfers abbrechen. Die Replizierungsbeziehung zwischen dem Ziel- und den Quell-Volumes unterbrechen, sodass

das Ziel-Volume zu Lese-/Schreibzugriff wird.

2. Mounten Sie das Volume, das den Kubernetes enthält, von der Ziel-SVM `etcd` Datendateien und Zertifikate auf dem Host, die als Master Node eingerichtet werden.
3. Kopieren Sie alle erforderlichen Zertifikate zum Kubernetes-Cluster unter `/etc/kubernetes/pki` Und das usw. `member` Dateien unter `/var/lib/etcd`.
4. Erstellen Sie einen Kubernetes-Cluster, indem Sie den ausführen `kubeadm init` Befehl mit dem `--ignore-preflight-errors=DirAvailable—var-lib-etcd` Flagge. Die Hostnamen sollten mit dem Quell-Kubernetes-Cluster übereinstimmen.
5. Führen Sie die aus `kubectl get crd` Befehl zur Überprüfung, ob alle benutzerdefinierten Trident Ressourcen aufgerufen wurden. Trident-Objekte werden abgerufen, um sicherzustellen, dass alle Daten verfügbar sind.
6. Bereinigen Sie die vorherigen Back-Ends und erstellen Sie mit Trident neue Back-Ends. Geben Sie die neue Management-LIF, den neuen SVM-Namen und das Passwort der Ziel-SVM an.

Disaster-Recovery-Workflow für persistente Applikations-Volumes

Beschreiben Sie in den folgenden Schritten, wie SnapMirror Ziel-Volumes bei einem Ausfall für Container-Workloads zur Verfügung gestellt werden können:

1. Beenden Sie alle geplanten SnapMirror-Transfers und beenden Sie alle laufenden SnapMirror Transfers. Die Replizierungsbeziehung zwischen dem Ziel- und dem Quell-Volume unterbrechen, sodass das Ziel-Volume zu Lese-/Schreibzugriff wird. Bereinigung der Bereitstellungen, für die PVC verwendet wurde, die an Volumes auf der Quell-SVM gebunden sind
2. Nachdem die Kubernetes-Cluster auf der Zielseite eingerichtet wurde, verwenden Sie die oben beschriebenen Schritte, um die Implementierungen, PVCs und PV aus dem Kubernetes-Cluster zu bereinigen.
3. Erstellen Sie auf Trident neue Back-Ends, indem Sie die neue Management- und Daten-LIF, den neuen SVM-Namen und das Passwort der Ziel-SVM angeben.
4. Importieren Sie die erforderlichen Volumes als PV, der an eine neue PVC gebunden ist, mithilfe der Trident-Importfunktion.
5. Implementieren Sie die Applikationsimplementierungen mithilfe der neu erstellten VES neu.

Daten mit Element Snapshots wiederherstellen

Sichern Sie die Daten auf einem Element-Volume, indem Sie einen Snapshot-Zeitplan für das Volume festlegen und sicherstellen, dass die Snapshots in den erforderlichen Intervallen erstellt werden. Der Snapshot-Zeitplan sollte mithilfe der Element UI oder APIs festgelegt werden. Derzeit ist es nicht möglich, einen Snapshot-Zeitplan auf ein Volume über das festzulegen `solidfire-san` Treiber.

Im Falle einer Beschädigung von Daten können Sie einen bestimmten Snapshot auswählen und das Volume manuell über die Element UI oder APIs zum Snapshot zurückwechseln. Hierdurch werden alle Änderungen an dem Volume zurückgesetzt, die seit der Erstellung des Snapshots vorgenommen wurden.

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 haben `tridentctl` Applikation.

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 managen `tridentctl`. Siehe "Hier" Um zu erfahren, wie Astra Trident CHAP auf ONTAP Back-Ends konfiguriert.

CHAP-Unterstützung für ONTAP-Back-Ends ist mit Trident 20.04 und höher verfügbar.

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 Trident als CSI-bereitstellung installiert wird, verwaltet es die CHAP-Geheimnisse und speichert sie in einem `tridentvolume` CR-Objekt für das jeweilige PV. Beim Erstellen eines PV verwendet CSI Astra Trident die CHAP-Schlüssel, um eine iSCSI-Sitzung zu initiieren und über CHAP mit dem NetApp HCI- und SolidFire-System zu kommunizieren.

Die von CSI Trident erstellten Volumes werden keiner Volume Access Group zugeordnet.

Im nicht-CSI-Frontend wird die Anbindung von Volumes als Geräte auf den Worker-Nodes durch Kubernetes übernommen. Nach der Volume-Erstellung ruft Astra Trident die API zum NetApp HCI/SolidFire System auf, um die Geheimnisse zu rufen, falls das Geheimnis für diesen Mandanten nicht bereits vorhanden ist. Astra Trident leitet die Geheimnisse an Kubernetes weiter. Das Kubelet, das sich auf jedem Node befindet, greift über die Kubernetes API auf die Geheimnisse zu und verwendet sie zum Ausführen/Aktivieren von CHAP zwischen jedem Node, der auf das Volume zugreift, und dem NetApp HCI/SolidFire System, in dem sich die Volumes befinden.

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 einstellen `true` 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-Verschlüsselungsfahne 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-Verschlüsselungsfahne auf setzen `false`.

- Sie können in Astra Trident manuell ein NVE-Volume erstellen, indem Sie explizit das NVE-Verschlüsselungsflag auf festlegen `true`.

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 Volumen 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.

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 angezeigt `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.

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 erstellenden 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 bereit. Das bewährte und umfangreiche Storage-Portfolio von NetApp umfasst sowohl in der Public Cloud als auch On-Premises. 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 von NetApp für mehrere Protokolle und bietet erweiterte Datenmanagement-Funktionen für alle Applikationen. 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 Kunden Filesysteme auf Basis des NetApp ONTAP Storage-Betriebssystems starten und ausführen können. Mit FSX für ONTAP können Kunden bereits bekannte NetApp Funktionen sowie deren Performance und Administration nutzen und gleichzeitig die Einfachheit, Agilität, Sicherheit und Skalierbarkeit beim Speichern von Daten in AWS nutzen. FSX für ONTAP unterstützt viele 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. Dieser Service ermöglicht die Ausführung aller Workloads, auch älterer Applikationen, in der GCP Cloud. 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
trident.netapp.io/fileSystem	Dateisystem	ontap-san, solidfire-san, ontap-san-Economy

Anmerkung	Volume-Option	Unterstützte Treiber
trident.netapp.io/cloneFromPVC	KlonSourceVolume	ontap-nas, ontap-san, solidfire-san, Azure-netapp-Dateien, gcp-cvs, 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 Löschaktion 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. Sie können mit diesen Annotationen ein PVC klonen, ohne dass die CSI-Implementierung (unter Kubernetes 1.13 und früher) verwendet werden muss oder wenn Ihre Kubernetes Version keine Beta Volume Snapshots (Kubernetes 1.16 und älter) unterstützt. Beachten Sie, dass Trident 19.10 den CSI-Workflow zum Klonen von einer PVC unterstützt.

Sie können das verwenden `cloneFromPVC` Und `splitOnClone` Anmerkungen mit CSI Trident sowie das traditionelle nicht-CSI-Frontend.

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 klont 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. In Trident Volume-Objekten finden Sie eine vollständige Beschreibung der Parameter und Einstellungen, die mit Trident Volumes verbunden sind.

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 Back-End-Namen zu Listen von Storage-Pools innerhalb
Zusätzlich StoragePools	Zuordnen[String]StringList	Nein	Zuordnung von Back-End-Namen zu Listen von Storage-Pools innerhalb
Unter Ausnahme von StoragePools	Zuordnen[String]StringList	Nein	Zuordnung von Back-End-Namen zu Listen von Storage-Pools innerhalb

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 angegeben	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 Bereitstellungsmethode	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 Klonen	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 kommagetrennte 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. xfsprogs.

Attribut	Typ	Werte	Beschreibung	Wichtige Faktoren	Kubernetes-Version
Fstype	Zeichenfolge	Ext4, ext3, xfs usw.	Der Filesystem-Typ 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

Attribut	Typ	Werte	Beschreibung	Wichtige Faktoren	Kubernetes-Version
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 einstellen `Retain` Bedeutet das `VolumeSnapshotContent` Und der physische Snapshot wird beibehalten.

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 Volume-Snapshot die Anforderung eines Benutzers, einen Snapshot eines vorhandenen PVC zu erstellen.

Sobald eine Volume Snapshot-Anfrage eingeht, managt Trident automatisch die Erstellung des Snapshots für das Volume auf dem Backend und legt den Snapshot offen, indem er einen eindeutigen erstellt `VolumeSnapshotContent` Objekt: Sie können Snapshots aus vorhandenen VES erstellen und die Snapshots als Datenquelle beim Erstellen neuer VES verwenden.

> Der Lebenszyklus eines VolumeSnapshots 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.

> Trident erstellt `VolumeSnapshotContent` Objekte werden beim Kubernetes Cluster automatisch auf Basis der Volumes registriert, die bereitgestellt werden. Sie sollten diese nicht selbst verwalten.

Der `VolumeSnapshotContent` Das 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 einen `VolumeSnapshotContent` Objekt-Storage erstellt und damit den Snapshot der Kubernetes API zur Verfügung gestellt.

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

Metadatenspeicher gespeichert. Dadurch ist kein separater Speicher für Trident erforderlich.

Ab Version 19.07 verwendet Trident mehrere Lösungen `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.
- Beim Upgrade von einer früheren Version von Trident (eine Version, die verwendet wurde `etcd` Um den Status beizubehalten) migriert das Trident-Installationsprogramm die Daten von dem `etcd` Schlüsselwert-Datenspeicher und Erstellung der entsprechenden CRD-Objekte.
- 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

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:

Trident Back-End-Objekte

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".

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.

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

Attribut	Typ	Erforderlich	Beschreibung
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
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 desinfiziert 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 des Bereitstellungsprozesses.

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
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 durch Abrufen der Details des Back-Volume gefüllt.

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]
```

Flaggen: * `-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 Telemetry (Standard „netapp/Trident AutoSupport:20.07.0“).
- `--autosupport-proxy string`: Die Adresse/der Port eines Proxy für den Versand von AutoSupport Telemetrie.
- `--csi`: CSI Trident installieren (Überschreiben nur für Kubernetes 1.13, erfordert Feature-Gates).
- `--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`: Überschreiben Sie die HTTP-Anforderung-Timeout für die REST-API des Trident-Controllers (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]
```

Flaggen: * `-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.

upgrade

Sie können die ausführen `upgrade` Befehle für das Upgrade einer Ressource in Astra Trident.

```
Usage:
tridentctl upgrade [option]
```

Verfügbare Option:
`volume`: Upgrade eines oder mehrerer persistenter Volumes von NFS/iSCSI auf CSI.

version

Sie können die ausführen `version` Flags zum Drucken der Version von `tridentctl` Und den Running Trident Service.

```
Usage:
  tridentctl version [flags]
```

Flaggen: * `--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. `iscsiadm` Managt iSCSI-Mounts und verwendet Host-Netzwerke für die Kommunikation mit dem iSCSI-Daemon.
Host-IPC	NFS nutzt Prozesskommunikation (IPC) mit dem NFSD.
Host-PID	Muss gestartet werden `rpc-statd` Für NFS. Astra Trident fragt die Host-Prozesse ab, um festzustellen, ob `rpc-statd` Wird vor dem Mounten von NFS-Volumes ausgeführt.
Sorgen	Der `SYS_ADMIN` Diese Funktion wird als Teil der Standardfunktionen für privilegierte Container bereitgestellt. Docker legt beispielsweise die folgenden Funktionen für privilegierte Container fest: `CapPrm: 0000003fffffffff` `CapEff: 0000003fffffffff`
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 `spc_t` („Super Privileged Container“)-Domain, und unprivilegierte Container werden im ausgeführt `container_t` Domäne. Ein `containerd`, Mit `container-selinux` Installiert, alle Container werden im ausgeführt `spc_t` Domain, die SELinux effektiv deaktiviert. Aus diesem Grund wird Astra Trident nicht hinzugefügt `seLinuxOptions` 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

`pod-security.kubernetes.io/enforce`
 	
`pod-security.kubernetes.io/enforce-version`
	Ermöglicht die Aufnahme der Trident Controller und Knoten im Namespace für die Installation. Ändern Sie nicht die Namespace-Bezeichnung.	`enforce: privileged`
		
		`enforce-version: <version of the current cluster or highest version of PSS tested.>`

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
`allowPrivilegeEscalation`	Privilegierte Container müssen die Eskalation von Berechtigungen ermöglichen.	`true`
`allowedCSIDrivers`	Trident verwendet keine kurzlebigen CSI-Inline-Volumes.	Leer
`allowedCapabilities`	Für Trident Container ohne Privilegien sind nicht mehr Funktionen erforderlich als für die Standardwerte. Privilegierte Container erhalten alle möglichen Funktionen.	Leer
`allowedFlexVolumes`	Trident verwendet kein a "FlexVolume-Treiber", Sie sind daher nicht in die Liste der zulässigen Volumen.	Leer
`allowedHostPaths`	Der Trident-Node-Pod hängt das Root-Dateisystem des Node zusammen, daher bietet es keinen Vorteil, diese Liste zu setzen.	Leer
`allowedProcMountTypes`	Trident verwendet keine `ProcMountTypes`.	Leer
`allowedUnsafeSysctls`	Trident erfordert keine Unsicherheit `sysctls`.	Leer
`defaultAddCapabilities`	Zu privilegierten Containern müssen keine Funktionen hinzugefügt werden.	Leer

Feld	Beschreibung	Standard
`defaultAllowPrivilegeEscalation`	In jedem Trident Pod werden Berechtigungen erteilt.	`false`
`forbiddenSysctls`	Nein `sysctls` Zulässig.	Leer
`fsGroup`	Trident Container werden als Root ausgeführt.	`RunAsAny`
`hostIPC`	Das Mounten von NFS-Volumes erfordert die Kommunikation zwischen dem Host IPC und dem `nfsd`	`true`
`hostNetwork`	Iscsiadm erfordert, dass das Hostnetzwerk mit dem iSCSI-Daemon kommunizieren kann.	`true`
`hostPID`	Host PID ist erforderlich, um zu überprüfen, ob `rpc-statd` Wird auf dem Node ausgeführt.	`true`
`hostPorts`	Trident verwendet keine Host Ports.	Leer
`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`	Iscsiadm 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`
`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`

Etiketten	Beschreibung	Standard
`seLinuxContext`	Trident ist nicht eingerichtet `seLinuxOptions` 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.
`volumes`	Trident Pods erfordern diese Volume-Plug-ins.	`hostPath, downwardAPI, projected, emptyDir`

Frühere Versionen der Dokumentation

Wenn Astra Trident 23.01 nicht ausgeführt wird, ist die Dokumentation zu früheren Versionen verfügbar.

- "Astra Trident 22.10"
- "Astra Trident 22.07"
- "Astra Trident 22.04"
- "Astra Trident 22.01"

Rechtliche Hinweise

Rechtliche Hinweise ermöglichen den Zugriff auf Copyright-Erklärungen, Marken, Patente und mehr.

Urheberrecht

"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/"

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/"

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 © 2025 NetApp. Alle Rechte vorbehalten. Gedruckt in den USA. Dieses urheberrechtlich geschützte Dokument darf ohne die vorherige schriftliche Genehmigung des Urheberrechtsinhabers in keiner Form und durch keine Mittel – weder grafische noch elektronische oder mechanische, einschließlich Fotokopieren, Aufnehmen oder Speichern in einem elektronischen Abrufsystem – auch nicht in Teilen, vervielfältigt werden.

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

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

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

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

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

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

Markeninformationen

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