Trident 26.02 Dokumentation

Trident

NetApp
August 04, 2026

This PDF was generated from https://docs.netapp.com/de-de/trident-2602/index.html on August 04, 2026.
Always check docs.netapp.com for the latest.

Inhalt

Trident 26.02 Dokumentation . . . 1
Versionshinweise . . . 2
Was ist neu . . . 2
Was ist neu in 26.02.1 . . . 2
Was ist neu in 26.02 . . . 3
Was ist neu in 25.10 . . . 5
Änderungen in 25.06.2 . . . 8
Änderungen in 25.06.1 . . . 8
Änderungen in 25.06 . . . 8
Änderungen in 25.02.1 . . . 10
Änderungen in 25.02 . . . 11
Änderungen in 24.10.1 . . . 12
Änderungen in 24.10 . . . 13
Änderungen in 24.06 . . . 14
Änderungen in 24.02 . . . 15
Änderungen in 23.10 . . . 16
Änderungen in 23.07.1 . . . 16
Änderungen in 23.07 . . . 17
Änderungen in 23.04 . . . 18
Änderungen in 23.01.1 . . . 19
Änderungen in 23.01 . . . 19
Änderungen in 22.10 . . . 20
Änderungen in 22.07 . . . 21
Änderungen in 22.04 . . . 22
Änderungen in 22.01.1 . . . 23
Änderungen in 22.01.0 . . . 23
Änderungen in 21.10.1 . . . 24
Änderungen in 21.10.0 . . . 24
Bekannte Probleme . . . 25
Weitere Informationen . . . 26
Frühere Versionen der Dokumentation . . . 26
NetApp Trident-Unterstützung für ONTAP ASA r2-Speichersysteme . . . 27
Unterstützte Operationen . . . 27
Nicht unterstützte Operationen . . . 27
Bekannte Probleme . . . 28
VolumeSnapshots erreicht nicht den ReadyToUse-Zustand . . . 28
Das Wiederherstellen von Restic-Backups großer Dateien kann fehlschlagen . . . 28
Los geht’s . . . 29
Erfahren Sie mehr über Trident . . . 29
Erfahren Sie mehr über Trident . . . 29
Trident-Architektur . . . 30
Konzepte . . . 32
Schnellstart für Trident . . . 36

Was kommt als Nächstes? 37
Anforderungen 37
Wichtige Informationen zu Trident 37
Unterstützte Frontends (Orchestratoren) 37
Unterstützte Backends (Storage) 38
Trident Unterstützung für KubeVirt und OpenShift Virtualisierung 38
Funktionsanforderungen 39
Getestete Host-Betriebssysteme 39
Hostkonfiguration 40
Speichersystemkonfiguration 40
Trident-Ports 40
Container-Images und zugehörige Kubernetes-Versionen 40
Trident installieren 41
Erfahren Sie mehr über die Trident-Installation 41
Wichtige Informationen zu Trident 26.02 41
Bevor Sie beginnen 41
Wählen Sie Ihre Installationsmethode 42
Wählen Sie Ihren Installationsmodus 44
Wählen Sie den Prozess basierend auf Ihrer Methode und Ihrem Modus aus 44
Wechsel zwischen Installationsmethoden 45
Weitere bekannte Konfigurationsoptionen 45
Installation mit dem Trident operator 45
Manuelle Bereitstellung des Trident Operators (Standardmodus) 45
Manuelle Bereitstellung des Trident Operators (Offline-Modus) 51
Trident Operator mit Helm bereitstellen (Standardmodus) 57
Trident operator mit Helm bereitstellen (Offline mode) 66
Trident-Operator-Installation anpassen 75
Installation mit tridentctl 87
Installation mit tridentctl 87
tridentctl-Installation anpassen 91
Installation mit dem OpenShift zertifizierten Operator 92
Installieren Sie Trident mit OpenShift OperatorHub 92
Wechseln Sie zum OpenShift zertifizierten Trident Operator 95
Verwenden Sie Trident 99
Bereiten Sie den Worker-Knoten vor 99
Die richtigen Werkzeuge auswählen 99
Knotendiensterkennung 99
NFS-Volumes 100
iSCSI-Volumes 100
NVMe/TCP Volumes 104
SCSI über FC-Volumes 105
Bereiten Sie die Bereitstellung von SMB-Volumes vor 108
Backends konfigurieren und verwalten 109
Backends konfigurieren 109
Azure NetApp Files 110

Google Cloud NetApp Volumes . . . 129
Konfigurieren Sie ein NetApp HCI- oder SolidFire-Backend . . . 156
ONTAP SAN-Treiber . . . 161
ONTAP NAS-Treiber . . . 192
Amazon FSx for NetApp ONTAP . . . 232
Backends mit kubectl erstellen . . . 271
Backends verwalten . . . 278
Speicherklassen erstellen und verwalten . . . 288
Erstellen Sie eine Speicherklasse . . . 288
Speicherklassen verwalten . . . 291
Volumes bereitstellen und verwalten . . . 293
Ein Volume bereitstellen . . . 293
Volumen erweitern . . . 297
Grenzen des RWX NVMe-Subsystems verstehen . . . 308
Controller-Skalierbarkeit . . . 310
Automatische Volumenerweiterung . . . 314
Autogrow-Richtlinien verwalten . . . 321
Importmengen . . . 330
Volumennamen und Beschriftungen anpassen . . . 342
Ein NFS-Volume über Namespaces hinweg freigeben . . . 345
Volumes über Namespaces hinweg klonen . . . 349
Volumes replizieren mit SnapMirror . . . 352
CSI-Topologie verwenden . . . 358
Mit Snapshots arbeiten . . . 366
Arbeiten mit Volume-Group-Snapshots . . . 374
Trident verwalten und überwachen . . . 379
Trident aktualisieren . . . 379
Trident aktualisieren . . . 379
Upgrade mit dem Operator . . . 380
Upgrade mit tridentctl . . . 385
Trident mit tridentctl verwalten . . . 386
Befehle und globale Flags . . . 386
Befehlsoptionen und Flags . . . 388
Plugin-Unterstützung . . . 394
Trident überwachen . . . 394
Überblick . . . 394
Schritt 1: Definieren Sie ein Prometheus-Ziel . . . 395
Schritt 2: Erstellen Sie einen Prometheus ServiceMonitor . . . 395
Schritt 3: Trident-Metriken mit PromQL abfragen . . . 397
Erfahren Sie mehr über Trident AutoSupport-Telemetrie . . . 398
Trident-Metriken deaktivieren . . . 399
Trident deinstallieren . . . 399
Ermitteln Sie die ursprüngliche Installationsmethode . . . 399
Entfernen einer Trident-Operatorinstallation . . . 399
Deinstallieren einer `tridentctl`Installation . . . 400

Trident für Docker . . . 401
Voraussetzungen für die Bereitstellung . . . 401
Überprüfen Sie die Anforderungen . . . 401
NVMe-Tools . . . 403
FC-Tools . . . 404
Trident bereitstellen . . . 406
Docker-verwaltete Plugin-Methode (Version 1.13/17.03 und später) . . . 406
Traditionelle Methode (Version 1.12 oder früher) . . . 408
Trident beim Systemstart starten . . . 410
Aktualisieren oder deinstallieren Sie Trident . . . 410
Upgrade . . . 411
Deinstallieren . . . 412
Arbeiten mit Volumes . . . 412
Erstellen Sie ein Volumen . . . 412
Ein Volume entfernen . . . 413
Ein Volume klonen . . . 413
Extern erstellte Volumes zugreifen . . . 415
Treiber-spezifische Volume-Optionen . . . 415
Protokolle sammeln . . . 421
Sammeln Sie Protokolle zur Fehlerbehebung . . . 421
Allgemeine Tipps zur Fehlerbehebung . . . 422
Mehrere Trident-Instanzen verwalten . . . 422
Schritte für das Docker-verwaltete Plugin (Version 1.13/17.03 oder später) . . . 422
Schritte für traditionelle (Version 1.12 oder älter) . . . 423
Speicherkonfigurationsoptionen . . . 423
Globale Konfigurationsoptionen . . . 423
ONTAP-Konfiguration . . . 424
Element-Softwarekonfiguration . . . 433
Bekannte Probleme und Einschränkungen . . . 435
Das Upgrade des Trident Docker Volume Plugin auf 20.10 und höher von älteren Versionen führt zu einem Upgrade-Fehler mit der Fehlermeldung no such file or directory. . . . 435
Volume-Namen müssen mindestens 2 Zeichen lang sein. . . . 436
Docker Swarm weist bestimmte Verhaltensweisen auf, die Trident daran hindern, es mit jeder Speicher- und Treiberkombination zu unterstützen. . . . 436
Wenn ein FlexGroup bereitgestellt wird, stellt ONTAP kein zweites FlexGroup bereit, wenn das zweite FlexGroup ein oder mehrere Aggregate mit dem FlexGroup, das bereitgestellt wird, gemeinsam hat. . . . 436
Bewährte Verfahren und Empfehlungen . . . 437
Bereitstellung . . . 437
In einem dedizierten Namensraum bereitstellen . . . 437
Verwenden Sie Quoten und Bereichsgrenzen, um den Speicherverbrauch zu kontrollieren . . . 437
Storage-Konfiguration . . . 437
Plattformübersicht . . . 437
ONTAP und Cloud Volumes ONTAP Best Practices . . . 438
SolidFire Best Practices . . . 442
Wo finde ich weitere Informationen? . . . 444

Trident integrieren . . . 445
Fahrerauswahl und -bereitstellung . . . 445
Speicherklassendesign . . . 447
Design des virtuellen Pools . . . 448
Volumenoperationen . . . 450
Metrikdienst . . . 453
Datenschutz und Notfallwiederherstellung . . . 454
Trident-Replikation und -Wiederherstellung . . . 454
SVM-Replikation und -Wiederherstellung . . . 455
Volumenreplikation und Wiederherstellung . . . 456
Snapshot-Datenschutz . . . 457
Automatisierung des Failovers von zustandsbehafteten Anwendungen mit Trident . . . 457
Details zum erzwungenen Abtrennen . . . 457
Details zum automatischen Failover . . . 458
Sicherheit . . . 463
Sicherheit . . . 463
Linux Unified Key Setup (LUKS) . . . 464
Kerberos-In-Flight-Verschlüsselung . . . 471
Schützen Sie Anwendungen mit Trident Protect . . . 480
Erfahren Sie mehr über Trident Protect . . . 480
Was kommt als Nächstes? . . . 480
Installieren Sie Trident Protect . . . 480
Trident Protect Anforderungen . . . 480
Trident Protect installieren und konfigurieren . . . 484
Installieren Sie das Trident Protect CLI-Plugin . . . 488
Trident Protect-Installation anpassen . . . 492
Trident Protect verwalten . . . 497
Trident Protect-Autorisierung und Zugriffskontrolle verwalten . . . 497
Überwachen Sie die Trident Protect-Ressourcen . . . 504
Generieren Sie ein Trident Protect Support-Bundle . . . 509
Trident Protect aktualisieren . . . 511
Anwendungen verwalten und schützen . . . 513
Verwenden Sie Trident Protect AppVault-Objekte, um Buckets zu verwalten . . . 513
Definieren Sie eine Anwendung für das Management mit Trident Protect . . . 527
Schützen Sie Anwendungen mit Trident Protect . . . 532
Anwendungen wiederherstellen . . . 545
Anwendungen mit NetApp SnapMirror und Trident Protect replizieren . . . 563
Anwendungen mit Trident Protect migrieren . . . 580
Trident Protect-Ausführungshooks verwalten . . . 584
Trident Protect deinstallieren . . . 596
Trident- und Trident Protect-Blogs . . . 597
Trident Blogs . . . 597
Trident Protect Blogs . . . 597
Wissen und Unterstützung . . . 599
Häufig gestellte Fragen . . . 599

Allgemeine Fragen . . . 599
Installieren und verwenden Sie Trident auf einem Kubernetes-Cluster . . . 599
Fehlerbehebung und Support . . . 600
Trident aktualisieren . . . 601
Backends und Volumes verwalten . . . 602
Fehlerbehebung . . . 606
Allgemeine Fehlerbehebung . . . 606
Fehlgeschlagene Trident-Bereitstellung mit dem Operator . . . 608
Fehlgeschlagene Trident-Bereitstellung mit `tridentctl` . . . 610
Trident und CRDs vollständig entfernen . . . 610
NVMe-Knoten-Unstaging-Fehler mit RWX-Raw-Block-Namespaces auf Kubernetes 1.26 . . . 611
NFSv4.2-Clients melden „ungültiges Argument“ nach dem Upgrade von ONTAP, wenn sie erwarten, dass „v4.2-xattrs“ aktiviert ist . . . 612
Support . . . 612
Trident Support-Lebenszyklus . . . 612
Selbsthilfe . . . 613
Community-Support . . . 613
NetApp technische Unterstützung . . . 613
Für weitere Informationen . . . 613
Referenz . . . 614
Trident-Ports . . . 614
Überblick . . . 614
Trident REST API . . . 616
Wann sollte die REST API verwendet werden? . . . 616
Verwendung der REST API . . . 617
Befehlszeilenoptionen . . . 617
Protokollierung . . . 617
Kubernetes . . . 618
Docker . . . 618
REST . . . 618
Kubernetes- und Trident-Objekte . . . 618
Wie interagieren die Objekte miteinander? . . . 619
Kubernetes `PersistentVolumeClaim`-Objekte . . . 619
Kubernetes `PersistentVolume`-Objekte . . . 621
Kubernetes `StorageClass`-Objekte . . . 621
Kubernetes `VolumeSnapshotClass`-Objekte . . . 625
Kubernetes `VolumeSnapshot`-Objekte . . . 625
Kubernetes `VolumeSnapshotContent`-Objekte . . . 626
Kubernetes `VolumeGroupSnapshotClass`-Objekte . . . 626
Kubernetes `VolumeGroupSnapshot`-Objekte . . . 627
Kubernetes `VolumeGroupSnapshotContent`-Objekte . . . 627
Kubernetes `CustomResourceDefinition`-Objekte . . . 628
Trident `StorageClass` Objekte . . . 628
Trident Backend-Objekte . . . 628

Trident `StoragePool` Objekte . . . 629
Trident `Volume` Objekte . . . 629
Trident `Snapshot` Objekte . . . 630
Trident `ResourceQuota` Objekt . . . 631
Pod-Sicherheitsstandards (PSS) und Security Context Constraints (SCC) . . . 632
Erforderlicher Kubernetes-Sicherheitskontext und zugehörige Felder . . . 633
Pod Security Standards (PSS) . . . 633
Pod-Sicherheitsrichtlinien (PSP) . . . 634
Security Context Constraints (SCC) . . . 635
Rechtliche Hinweise . . . 638
Copyright . . . 638
Marken . . . 638
Patente . . . 638
Datenschutzrichtlinie . . . 638
Open Source . . . 638

Trident 26.02 Dokumentation

Versionshinweise

Was ist neu

Die Versionshinweise enthalten Informationen zu neuen Funktionen, Verbesserungen und Fehlerbehebungen in der neuesten Version von NetApp Trident.

> Die `tridentctl`Binärdatei für Linux, die in der Installer-ZIP-Datei bereitgestellt wird, ist die getestete und unterstützte Version. Beachten Sie, dass die `macos`Binärdatei, die im `/extras`Teil der ZIP-Datei bereitgestellt wird, nicht getestet oder unterstützt wird.

Was ist neu in 26.02.1

Erfahren Sie, was neu ist in NetApp Trident und Trident Protect, einschließlich Verbesserungen, Fehlerbehebungen und veralteter Funktionen.

Trident

Korrekturen

- **Kubernetes**:
- Das Verhalten von Google Cloud NetApp Volumes (GCNV) SAN wurde korrigiert, um Flex Unified ZONAL-Pools für Block-Workloads (iSCSI) zu unterstützen. Sowohl zonale als auch regionale Konfigurationen werden nun unterstützt.
- Die Handhabung von Google Cloud NetApp Volumes (GCNV) NAS UNIFIED Pools wurde behoben, indem die Volume-Erstellung, Validierung, Benennung und die Suche nach Ausweichvolumes verbessert wurden, wenn `InternalID` fehlt.
- Es wurde ein Problem behoben, bei dem iSCSI-Gerätegrößenänderungen fehlschlagen konnten, wenn verfügbare Pfade und Portale nicht übereinstimmten.
- Es wurde ein zeitweise auftretendes Problem behoben, bei dem `blkid` der Dateisystemtyp für LUKS-Geräte falsch identifiziert wurde, wodurch die Erstellung neuer Volumes verhindert wurde.
- Es wurde ein Problem behoben, bei dem LUKS-Veröffentlichungs- und Staging-Vorgänge in großem Umfang den Trident Controller blockieren konnten. Benutzer müssen nun die mit den Volumes verknüpften LUKS-Passphrasen nachverfolgen.
- Es wurde ein Problem behoben, bei dem iSCSI-Multipath-Partitionen als Geistergeräte angezeigt werden konnten, wodurch CSI `NodeUnstage`-Operationen nicht erfolgreich ausgeführt werden konnten.
- Behoben wurde eine Race Condition im NVMe/TCP-Namespace bei gleichzeitiger Volume-Erstellung (Issue #1089).
- Gleichzeitige Klon-Snapshot-Operationen wurden behoben, wenn Snapshots innerhalb derselben Sekunde ausgeführt wurden.
- Es wurde ein Problem behoben, das verhinderte, dass KubeVirt VMs mit `dataVolumes` Trident Automated Workload Failover ein Failover durchführen konnten.
- Es wurde ein Fehler behoben, der beim Löschen einer ungültigen `TridentMirrorRelationship`benutzerdefinierten Ressource bei Verwendung des Trident concurrent core auftrat.
- Einrückungsproblem in der `tridentactionmirrorupdates` Kubernetes CRD YAML-Definition behoben

(Problem #1120).

- Der asynchrone Löschaufruf der REST-Qtree-API wurde behoben (Problem #1121).
- Behoben `ControllerPublish`, sodass nun der Dateisystemtyp der Volume-Konfiguration anstelle eines falschen Standard-Dateisystemtyps verwendet wird.
- Die Protokollierung beim Start des Controllers wurde korrigiert, sodass der Status gleichzeitig laufender Treiber nun korrekt gemeldet wird.
- Behobene Handhabung von angehaltenen Backends in asynchronem Trident.
- Erweiterte AWS-ARN-Verarbeitung zur Unterstützung spezieller Regionen.

Trident Protect

Korrekturen

Trident Protect-Abhängigkeiten wurden aktualisiert, um Sicherheitslücken zu beheben:

- gRPC-Go: "CVE-2026-33186"
- Go x/crypto SSH: "CVE-2024-45337"
- OpenTelemetry-Go: "CVE-2026-24051"
- Go-Laufzeit: "CVE-2025-22871"
- Zusätzliche Abhängigkeiten: "CVE-2025-23109"

Was ist neu in 26.02

Erfahren Sie, was neu ist in NetApp Trident und Trident Protect, einschließlich Verbesserungen, Fehlerbehebungen und veralteter Funktionen.

Trident

Verbesserungen

- **Kubernetes**:
  - Allgemeine Verfügbarkeit (GA) der Unterstützung für Parallelverarbeitung für ONTAP-NAS (nur NFS), ONTAP-SAN (iSCSI, FCP, NVMe) und Google Cloud NetApp Volumes (GCNV)-Treiber hinzugefügt, wodurch diese Funktionalität aus der Tech Preview übernommen wurde. Siehe "Dokumentation zur Skalierbarkeit des Controllers" für weitere Informationen.
  - Unterstützung für das automatische Volume-Wachstum in Trident basierend auf benutzerdefinierten Trident AutoGrow-Richtlinien wurde hinzugefügt. Siehe "Dokumentation zur automatischen Volumenerweiterung" für weitere Informationen.
  - Verbesserte Trident-Knotenkonkurrenz zur Optimierung der Skalierbarkeit von Knotenoperationen für NVMe-Volumes. Siehe "NVMe volumes Dokumentation" für weitere Informationen.
  - Unterstützung für das automatische Tiering von GCNV NAS-Volumes über PVC-Annotationen hinzugefügt (`tieringPolicy`, `tieringMinimumCoolingDays`, einschließlich Poolauswahl und Klonvererbung. Siehe "Konfigurieren Sie das automatische Tiering für die Google Cloud NetApp Volumes Dokumentation" für weitere Informationen.
  - Unterstützung für GCNV SAN Block (iSCSI) Volumes mit dem `google-cloud-netapp-volumes-san` Treiber hinzugefügt, einschließlich Bereitstellung, Hostgruppen-Zuordnung pro Node für LUN-Zugriff und Klon-von-Volume-Operationen. Siehe "Google Cloud NetApp Volumes Blockkonfigurationsdokumentation" für weitere Informationen.

- Unterstützung für die automatische Backend-Konfiguration für Amazon FSx for NetApp ONTAP hinzugefügt. Wenn Sie eine StorageClass erstellen, die die erforderlichen Parameter enthält, erstellt Trident automatisch das entsprechende Backend und die VolumeSnapshotClass, falls erforderlich. Siehe "Amazon FSx for NetApp ONTAP Backend-Konfigurationsdokumentation" für weitere Informationen.
- Es wurde Unterstützung für verschiedene Microsoft Azure-Clouds hinzugefügt, wie Azure Government und Azure China, sowie eine benutzerdefinierte Cloud-Konfiguration für Azure NetApp Files Backends. Siehe "Azure NetApp Files Backend-Konfigurationsdokumentation" für weitere Informationen.
- Unterstützung für Kubernetes 1.35 hinzugefügt. Siehe "Anforderungsdokumentation" für weitere Informationen.

Experimentelle Verbesserungen

Nicht für den Einsatz in Produktionsumgebungen.

- **[Tech Preview]**: Unterstützung für Parallelverarbeitung für ONTAP-NAS-Economy und ONTAP-SAN-Economy Treiber hinzugefügt.

Bekannte Probleme

Es gibt ein bekanntes Problem mit dem `csi-snapshotter` Sidecar. In allen Kubernetes-Versionen verhindert VolumeGroupSnapshots `v1beta1`, dass VolumeSnapshots den `ReadyToUse` Zustand erreichen.

Zwei Workarounds sind verfügbar:

1. Löschen Sie die VolumeGroupSnapshots CRDs, um VolumeGroupSnapshots zu deaktivieren, und installieren Sie dann Trident neu.
2. Installieren Sie VolumeGroupSnapshots `v1beta2` und `snapshot-controller` Version 8.4.0 oder höher und installieren Sie dann Trident erneut. VolumeGroupSnapshots funktionieren auf Kubernetes-Versionen vor v1.34 nicht.

Korrekturen

- **Kubernetes**:
  - Es wurde ein Problem behoben, bei dem durch das Aufheben der Veröffentlichung eines schreibgeschützten Klons Exportrichtlinienregeln vom Quellvolume in ONTAP-NAS-, ONTAP-NAS-Economy- und Google Cloud NetApp Volumes-Treibern entfernt wurden "Problem #1086".
  - Switched kubectl-Images auf leichte, Alpine-basierte Varianten, um Pull-Fehler nach der Abschaffung der öffentlichen Images durch Bitnami zu verhindern "Problem #1080".
  - Behebung der Beibehaltung der Annotation bestehender Bereitstellungen während des Trident-Upgrades "Problem #1004".
  - Klonen über verschiedene Speicherklassen hinweg zulassen, wenn beide Speicherklassen auf dasselbe Backend verweisen "Problem #1104".
  - Behebung von Fehlern bei der Knotenvorbereitung aufgrund von Zeitüberschreitungen in Cloud-Umgebungen mit Netzwerklatenzen. Erhöhte Zeitüberschreitungswerte für Cloud-basierte Installationen.
  - Es wurde ein Problem bei der LUN-Erstellung behoben, das dazu führte, dass das Dateisystemtyp-Attribut nicht gesetzt blieb, wenn der Prozess in den Wiederholungszustand wechselte.

- Die REST-API-Volume-Abfrage wurde so angepasst, dass der Volume-Status ignoriert wird, wodurch falsch negative Ergebnisse bei Volume-Abfragen vermieden werden.
- Verbesserte Effizienz des Trident-Controllers für den ontap-nas-economy-Treiber bei großflächiger Nutzung.
- Setzen Sie internalID während des LUN-Imports im ontap-san-economy-Treiber.
- Die Abfragelimits für Azure Resource Graph wurden erhöht, um mehr Subnetze zu verarbeiten.
- Verbesserte CSI- und ONTAP-Klon-Teilung-Zeitüberschreitungen, um Race Conditions mit einigen Backup-Anwendungen zu vermeiden "Problem #1098", "Problem #1100".
- Die Unterdrückung von LUKS-Fehlermeldungen "Problem #1069" wurde behoben.
- Die Behandlung veralteter LUKS-Mapper für sowohl iSCSI- als auch NVMe-Protokolle wurde korrigiert. Die verbesserte Bereinigungslogik verhindert Mount-Fehler durch verwaiste Geräte-Mapper.

- Feste Skalierungsbeschränkungen für RWX NVMe-Volumes.
- Das OpenTelemetry-Go-Paket wurde aktualisiert, um "CVE-2026-24051" zu beheben.

Trident Protect

Verbesserungen

- Trident Protect deaktiviert nun automatisch Schutzpläne und laufende Vorgänge vor einer Wiederherstellung vor Ort und aktiviert sie nach Abschluss der Wiederherstellung wieder. Weitere Informationen finden Sie unter "Anwendungen mit Trident Protect wiederherstellen".
- Das `runImmediately` Feld wurde dem Schedule-CR und das `--run-immediately` CLI-Flag hinzugefügt, um beim Erstellen eines Zeitplans eine sofortige Sicherung oder einen Snapshot auszulösen. Weitere Informationen finden Sie unter "Erstellen Sie einen Zeitplan für die Datensicherung".
- Unterstützung für die Angabe eines benutzerdefinierten Namens für die wiederhergestellte Anwendung mithilfe des `destinationApplicationName` Felds im Restore-CR oder des `--destination-app-name` CLI-Flags hinzugefügt. Weitere Informationen finden Sie unter "Anwendungen mit Trident Protect wiederherstellen".

Korrekturen

- Behobene Wiederherstellungsfehler, die dadurch verursacht wurden, dass Pods erstellt wurden, bevor ihre erforderlichen Servicekonten verfügbar waren.
- Behoben `Roles` und `RoleBindings` wurden bei der Wiederherstellung von Anwendungen übersprungen.
- Behoben, dass der ursprüngliche Clustername in der `tridentctl-protect get appvaultcontent` Ausgabe trotz korrekter Konfiguration nicht angezeigt wurde.
- Behobene Kopia-Wiederherstellungsfehler, die aufgrund fehlender `pipefail`-Handhabung ignoriert wurden.
- Snapshot- und Backup-Fehler behoben, die durch Ressourcenfilter verursacht wurden, die persistente Volumes ausschlossen.
- Falsche Wiederherstellung von PVCs in Multi-Namespace-Anwendungen mit identisch benannten PVCs über verschiedene Namensräume hinweg behoben, was zu Datenverlust führen konnte.

Was ist neu in 25.10

Erfahren Sie, was neu ist in Trident und Trident Protect, einschließlich Verbesserungen, Fehlerbehebungen und Außerkraftsetzungen.

Trident

Verbesserungen

- **Kubernetes**:
  - Unterstützung für CSI Volume Group Snapshots mit v1beta1 Volume Group Snapshot Kubernetes APIs für ONTAP-NAS NFS und ONTAP-SAN-Economy Treiber wurde zusätzlich zu ONTAP-SAN (iSCSI und FC) hinzugefügt. Siehe "Arbeiten mit Volume-Group-Snapshots".
  - Unterstützung für automatisches Workload-Failover mit erzwungener Volume-Trennung für ONTAP-NAS und ONTAP-NAS-Economy (ohne SMB in beiden NAS-Treibern) sowie für ONTAP-SAN und ONTAP-SAN-Economy Treiber hinzugefügt. Siehe "Automatisierung des Failovers von zustandsbehafteten Anwendungen mit Trident".
  - Verbesserte Trident-Knotenkonkurrenz für höhere Skalierbarkeit bei Knotenoperationen für FCP Volumes.
  - ONTAP AFX-Unterstützung für den ONTAP NAS-Treiber hinzugefügt. Siehe "ONTAP NAS-Konfigurationsoptionen und Beispiele".
  - Unterstützung für die Konfiguration von CPU- und Speicherressourcenanforderungen und -grenzen für Trident-Container über TridentOrchestrator CR und Helm-Chart-Werte hinzugefügt. ("Problem #1000", "Problem #927", "Problem #853", "Problem #592", "Problem #110").
  - FC-Unterstützung für ASAr2-Persönlichkeit hinzugefügt. Siehe "ONTAP SAN-Konfigurationsoptionen und Beispiele".
  - Es wurde eine Option hinzugefügt, um Prometheus-Metriken mit HTTPS anstelle von HTTP bereitzustellen. Siehe "Trident überwachen".
  - Eine Option `--no-rename` wurde hinzugefügt, um beim Importieren eines Volumes den ursprünglichen Namen beizubehalten, aber Trident die Verwaltung des Lebenszyklus des Volumes zu überlassen. Siehe "Importmengen".
  - Die Trident-Bereitstellung läuft jetzt in der Prioritätsklasse system-cluster-critical.
- Eine Option wurde hinzugefügt, damit der Trident Controller Host-Netzwerkfunktionen über helm, operator und tridentctl ("Problem #858") nutzen kann.
- Dem ANF-Treiber wurde die manuelle QoS-Unterstützung hinzugefügt, wodurch er in Trident 25.10 produktionsreif ist; diese experimentelle Erweiterung wurde in Trident 25.06 eingeführt.

Experimentelle Verbesserungen

ⓘ Nicht für den Einsatz in Produktionsumgebungen.

- **[Technische Vorschau]**: Unterstützung für Parallelverarbeitung für ONTAP-NAS (nur NFS) und ONTAP-SAN (NVMe für unified ONTAP 9) wurde zusätzlich zur bestehenden Technischen Vorschau für den ONTAP-SAN-Treiber (iSCSI- und FCP-Protokolle in unified ONTAP 9) hinzugefügt.

Korrekturen

- **Kubernetes**:
  - Die Namensinkonsistenz des CSI node-driver-registrar-Containers wurde behoben, indem Linux DaemonSet auf node-driver-registrar standardisiert wurde, um der Windows DaemonSet und der Container-Image-Namensgebung zu entsprechen.
  - Ein Problem wurde behoben, bei dem Exportrichtlinien für Legacy-Qtrees nicht ordnungsgemäß aktualisiert wurden.

- **Openshift**:
  - Behoben: Trident-Node-Pod startete auf Windows-Nodes in Openshift nicht, weil SCC allowHostDirVolumePlugin auf false gesetzt war ("Problem #950").
- Behoben: Kubernetes API QPS konnte nicht über Helm festgelegt werden ("Problem #975").
- Behoben: Unfähigkeit, einen Persistent Volume Claim (PVC), der auf einem Snapshot eines NVMe-basierten XFS-Dateisystem-PVC basiert, auf demselben Kubernetes-Knoten einzuhängen.
- Problem mit der Änderung der UUID nach einem Neustart des Hosts/Docker im NDVP-Modus behoben, indem eindeutige/gemeinsame Subsystemnamen pro Backend hinzugefügt wurden (z. B. netappdvp_subsystem).
- Behobene Mount-Fehler für iSCSI-Volumes während des Trident-Upgrades von Versionen vor 23.10 auf 24.10 und höher, wodurch das „invalid SANType“-Problem behoben wurde.
- Behobenes Problem, bei dem der Trident-Backend-Status nicht ohne Neustart des Trident Controllers auf online/offline wechselte.
- Intermittierender Race-Condition-Fehler behoben, der eine langsame PVC-Größenänderung verursachte.
- Behoben, dass Snapshots bei Fehlern beim Klonen von Volumes nicht bereinigt wurden.
- Fehler beim Freigeben eines Volumes behoben, wenn sein Gerätepfad vom Kernel geändert wurde.
- Fehler beim Entfernen des Volumes aus der Bereitstellungsphase aufgrund eines bereits geschlossenen LUKS-Geräts behoben.
- Problem behoben, bei dem langsame Speichervorgänge zu ContextDeadline-Fehlern führten.
- Trident Operator wartet für den konfigurierbaren k8s-Zeitüberschreitung, um die Trident Version zu überprüfen.

Trident Protect

NetApp Trident Protect bietet erweiterte Funktionen für das Management von Anwendungsdaten, die die Funktionalität und Verfügbarkeit zustandsbehafteter Kubernetes-Anwendungen verbessern, die durch NetApp ONTAP-Speichersysteme und den NetApp Trident CSI Storage Provisioner unterstützt werden.

Verbesserungen

- Anmerkungen hinzugefügt, um Snapshot-CR-Zeitüberschreitungen für geplante und Backup-CRs zu steuern:
  - `protect.trident.netapp.io/snapshot-completion-timeout`
  - `protect.trident.netapp.io/volume-snapshots-ready-to-use-timeout`
  - `protect.trident.netapp.io/volume-snapshots-created-timeout`

    Siehe "Unterstützte Backup- und Zeitplananmerkungen".
- Dem Schedule-CR wurde eine Anmerkung hinzugefügt, um die PVC-Bindungs-Zeitüberschreitung zu konfigurieren, die vom Backup-CR verwendet wird: `protect.trident.netapp.io/pvc-bind-timeout-sec`. Siehe "Unterstützte Backup- und Zeitplananmerkungen".
- Verbesserte `tridentctl-protect` Backup- und Snapshot-Listen mit einem neuen Feld zur Anzeige von Ausführungs-Hook-Fehlern.

Änderungen in 25.06.2

Trident

Korrekturen

- **Kubernetes**: Kritisches Problem behoben, bei dem beim Trennen von Volumes von Kubernetes-Knoten falsche iSCSI-Geräte erkannt wurden.

Änderungen in 25.06.1

Trident

> Für Kunden, die SolidFire verwenden, nicht auf 25.06.1 aktualisieren, da es ein bekanntes Problem beim Ausblenden von Volumes gibt. 25.06.2 wird in Kürze veröffentlicht, um dieses Problem zu beheben.

Korrekturen

- **Kubernetes**:
  - Es wurde ein Problem behoben, bei dem NQNs nicht überprüft wurden, bevor sie aus Subsystemen unzugeordnet wurden.
  - Ein Problem wurde behoben, bei dem mehrere Versuche, ein LUKS-Gerät zu schließen, zu Fehlern beim Trennen von Volumes führten.
  - Behoben: iSCSI-Volume-Unstage, wenn sich der Gerätepfad seit der Erstellung geändert hat.
  - Blockklonierung von Volumes über Speicherklassen hinweg.
- **OpenShift**: Es wurde ein Problem behoben, bei dem die iSCSI-Knotenvorbereitung mit OCP 4.19 fehlschlug.
- Die Zeitüberschreitung beim Klonen eines Volumes mit SolidFire Backends wurde erhöht ("Problem #1008").

Änderungen in 25.06

Trident

Verbesserungen

- **Kubernetes**:
  - Unterstützung für CSI Volume Group Snapshots mit `v1beta1` Volume Group Snapshot Kubernetes APIs für ONTAP-SAN iSCSI driver hinzugefügt. Siehe "Arbeiten mit Volume-Group-Snapshots".

    > VolumeGroupSnapshot ist eine Beta-Funktion in Kubernetes mit Beta-APIs. Kubernetes 1.32 ist die Mindestversion, die für VolumeGroupSnapshot erforderlich ist.

  - Unterstützung für ONTAP ASA r2 für NVMe/TCP zusätzlich zu iSCSI hinzugefügt. Siehe "ONTAP SAN-Konfigurationsoptionen und Beispiele".
  - Sichere SMB-Unterstützung für ONTAP-NAS und ONTAP-NAS-Economy Volumes hinzugefügt. Active Directory Benutzer und Gruppen können nun mit SMB Volumes für erhöhte Sicherheit verwendet werden. Siehe "Sichere SMB-Verbindungen aktivieren".

- Verbesserte Trident-Knotenkonkurrenz für höhere Skalierbarkeit bei Knotenoperationen für iSCSI-Volumes.
- Hinzugefügt `--allow-discards` beim Öffnen von LUKS-Volumes, um discard/TRIM-Befehle zur Speicherplatzrückgewinnung zu ermöglichen.
- Verbesserte Leistung beim Formatieren von LUKS-verschlüsselten Datenträgern.
- Erweiterte LUKS-Bereinigung für fehlgeschlagene, aber teilweise formatierte LUKS-Geräte.
- Verbesserte Trident-Knoten-Idempotenz für das Anhängen und Trennen von NVMe-Volumes.
- Added `internalID`-Feld zur Trident-Volume-Konfiguration für den ONTAP-SAN-Economy-Treiber.
- Unterstützung für die Volume-Replikation mit SnapMirror für NVMe-Backends hinzugefügt. Siehe "Volumes replizieren mit SnapMirror".

Experimentelle Verbesserungen

Nicht für den Einsatz in Produktionsumgebungen.

- [Technische Vorschau] Gleichzeitige Trident-Controller-Operationen wurden über das `--enable-concurrency` Feature-Flag aktiviert. Dadurch können Controller-Operationen parallel ausgeführt werden, was die Leistung in stark ausgelasteten oder großen Umgebungen verbessert.

  Diese Funktion ist experimentell und unterstützt derzeit eingeschränkte parallele Arbeitsabläufe mit dem ONTAP-SAN-Treiber (iSCSI und FCP Protokollen).

- [Tech Preview] Manuelle QOS-Unterstützung mit dem ANF-Treiber hinzugefügt.

Korrekturen

- **Kubernetes**:
  - Ein Problem mit CSI NodeExpandVolume wurde behoben, bei dem Multipath-Geräte inkongruente Größen aufweisen konnten, wenn die zugrunde liegenden SCSI-Festplatten nicht verfügbar waren.
  - Fehler beim Bereinigen doppelter Exportrichtlinien für ONTAP-NAS- und ONTAP-NAS-Economy-Treiber behoben.
  - Behoben: GCNV-Volumes verwendeten standardmäßig NFSv3, wenn `nfsMountOptions` nicht gesetzt war; jetzt werden sowohl NFSv3- als auch NFSv4-Protokolle unterstützt. Wenn `nfsMountOptions` nicht angegeben ist, wird die standardmäßige NFS-Version des Hosts (NFSv3 oder NFSv4) verwendet.
  - Behobenes Bereitstellungsproblem bei der Installation von Trident mit Kustomize ("Problem #831").
  - Fehlende Exportrichtlinien für aus Snapshots erstellte PVCs wurden behoben ("Problem #1016").
  - Behobenes Problem, bei dem die ANF-Volume-Größen nicht automatisch auf 1-GiB-Schritte ausgerichtet wurden.
  - Behobenes Problem bei der Verwendung von NFSv3 mit Bottlerocket.
- Behobenes Problem mit ONTAP-NAS-Economy-Volumes, die trotz fehlgeschlagener Größenänderungen auf bis zu 300 TB erweitert wurden.
- Behobenes Problem, bei dem Klon-Teilung bei Verwendung der ONTAP REST API synchron durchgeführt wurden.

Veraltete Funktionen:

- **Kubernetes**: Mindestunterstützte Kubernetes-Version auf v1.27 aktualisiert.

Trident Protect

NetApp Trident Protect bietet erweiterte Funktionen für das Management von Anwendungsdaten, die die Funktionalität und Verfügbarkeit zustandsbehafteter Kubernetes-Anwendungen verbessern, die durch NetApp ONTAP-Speichersysteme und den NetApp Trident CSI Storage Provisioner unterstützt werden.

Verbesserungen

- Verbesserte Wiederherstellungszeiten, mit der Option, häufigere vollständige Backups durchzuführen.
- Verbesserte Granularität der Anwendungsdefinition und selektive Wiederherstellung mit Group-Version-Kind (GVK)-Filterung.
- Effiziente Resynchronisierung und umgekehrte Replikation bei Verwendung von AppMirrorRelationship (AMR) mit NetApp SnapMirror, um eine vollständige PVC-Replikation zu vermeiden.
- Funktion hinzugefügt, EKS Pod Identity zu verwenden, um AppVault-Buckets zu erstellen, wodurch die Notwendigkeit entfällt, ein Secret mit den Bucket-Anmeldeinformationen für EKS-Cluster anzugeben.
- Die Möglichkeit wurde hinzugefügt, das Wiederherstellen von Labels und Annotationen im Restore-Namespace bei Bedarf zu überspringen.
- AppMirrorRelationship (AMR) prüft nun die Erweiterung des Quell-PVC und führt bei Bedarf die entsprechende Erweiterung am Ziel-PVC durch.

Korrekturen

- Behobener Fehler, bei dem Snapshot-Anmerkungswerte aus vorherigen Snapshots auf neuere Snapshots angewendet wurden. Alle Snapshot-Anmerkungen werden jetzt korrekt angewendet.
- Standardmäßig ein Geheimnis für die Datenübertragungsverschlüsselung (Kopia / Restic) definiert, falls nicht definiert.
- Verbesserte Validierung und Fehlermeldungen für die Erstellung von S3 appvault hinzugefügt.
- AppMirrorRelationship (AMR) repliziert PVs jetzt nur noch im Zustand "Bound", um Fehlversuche zu vermeiden.
- Behobenes Problem, bei dem Fehler angezeigt wurden, wenn AppVaultContent auf einem AppVault mit einer großen Anzahl von Backups abgerufen wurde.
- KubeVirt VMSnapshots sind von Wiederherstellungs- und Failover-Vorgängen ausgeschlossen, um Fehler zu vermeiden.
- Behobenes Problem mit Kopia, bei dem Snapshots vorzeitig entfernt wurden, weil der Kopia-Standardaufbewahrungsplan die vom Benutzer im Zeitplan festgelegten Einstellungen überschrieben hat.

Änderungen in 25.02.1

Trident

Korrekturen

- **Kubernetes**:
  - Es wurde ein Problem im Trident-Operator behoben, bei dem die Namen und Versionen der Sidecar-Images falsch angezeigt wurden, wenn eine nicht standardmäßige Image-Registry verwendet wurde

("Problem #983").

- Das Problem wurde behoben, bei dem Multipath-Sitzungen während eines ONTAP Failover-Givebacks nicht wiederhergestellt werden konnten ("Problem #961").

Änderungen in 25.02

Ab Trident 25.02 enthält die Zusammenfassung „Neuerungen“ Details zu Verbesserungen, Fehlerbehebungen und Veralterungen sowohl für Trident als auch für Trident Protect Releases.

Trident

Verbesserungen

- **Kubernetes**:
  - Unterstützung für ONTAP ASA r2 für iSCSI hinzugefügt.
  - Unterstützung für das erzwungene Trennen von ONTAP-NAS-Volumes bei nicht ordnungsgemäßem Herunterfahren von Knoten wurde hinzugefügt. Neue ONTAP-NAS-Volumes verwenden nun volumespezifische Exportrichtlinien, die von Trident verwaltet werden. Ein Upgrade-Pfad für bestehende Volumes wurde bereitgestellt, um beim Aufheben der Veröffentlichung auf das neue Exportrichtlinienmodell umzustellen, ohne aktive Workloads zu beeinträchtigen.
  - Anmerkung cloneFromSnapshot hinzugefügt.
  - Unterstützung für das Klonen von Volumes über Namespaces hinweg hinzugefügt.
  - Erweiterte iSCSI-Selbstreparatur-Scan-Behebungen zur Initiierung von erneuten Scans nach exakt Host, Kanal, Ziel und LUN-ID.
  - Unterstützung für Kubernetes 1.32 hinzugefügt.
- **OpenShift**:
  - Unterstützung für die automatische iSCSI-Knotenvorbereitung für RHCOS auf ROSA-Clustern hinzugefügt.
  - Unterstützung für OpenShift Virtualization für ONTAP-Treiber hinzugefügt.
- Fibre Channel-Unterstützung im ONTAP-SAN-Treiber hinzugefügt.
- NVMe LUKS-Unterstützung hinzugefügt.
- Für alle Basisimages wurde auf ein scratch image umgestellt.
- iSCSI-Verbindungsstatus-Erkennung und Protokollierung hinzugefügt, wenn iSCSI-Sitzungen protokolliert sein sollten, es aber nicht sind ("Problem #961").
- Unterstützung für SMB-Volumes mit dem google-cloud-netapp-volumes Treiber hinzugefügt.
- Unterstützung hinzugefügt, damit ONTAP Volumes die Wiederherstellungswarteschlange beim Löschen überspringen können.
- Unterstützung hinzugefügt, um Standardbilder mithilfe von SHAs anstelle von Tags zu überschreiben.
- Dem tridentctl Installer wurde das Flag image-pull-secrets hinzugefügt.

Korrekturen

- **Kubernetes**:
  - Fehlende Knoten-IP-Adressen aus automatischen Exportrichtlinien wurden korrigiert ("Problem #965").
  - Automatisches Umschalten der Exportrichtlinien auf volumenbasierte Richtlinien für ONTAP-NAS-

Economy wurde behoben.

- Backend-Konfigurations-Zugangsdaten korrigiert, um alle verfügbaren AWS-ARN-Partitionen zu unterstützen ("Problem #913").
- Option hinzugefügt, um die automatische Konfigurator-Abstimmung im Trident Operator zu deaktivieren ("Problem #924").
- Hinzugefügt securityContext für den csi-resizer-Container ("Problem #976").

Trident Protect

NetApp Trident Protect bietet erweiterte Funktionen für das Management von Anwendungsdaten, die die Funktionalität und Verfügbarkeit zustandsbehafteter Kubernetes-Anwendungen verbessern, die durch NetApp ONTAP-Speichersysteme und den NetApp Trident CSI Storage Provisioner unterstützt werden.

Verbesserungen

- Backup- und Wiederherstellungsunterstützung für KubeVirt / OpenShift Virtualisierungs-VMs wurde für beide volumeMode: File und volumeMode: Block (Rohgerät) Speicher hinzugefügt. Diese Unterstützung ist mit allen Trident-Treibern kompatibel und verbessert die bestehenden Schutzfunktionen bei der Replikation von Speicher mit NetApp SnapMirror und Trident Protect.
- Die Möglichkeit, das Einfrieren auf Anwendungsebene in Kubevirt-Umgebungen zu steuern, wurde hinzugefügt.
- Unterstützung für die Konfiguration von AutoSupport-Proxy-Verbindungen hinzugefügt.
- Die Möglichkeit wurde hinzugefügt, ein Geheimnis für die Datenübertragungsverschlüsselung (Kopia / Restic) zu definieren.
- Die Möglichkeit, einen Ausführungshook manuell auszuführen, wurde hinzugefügt.
- Die Möglichkeit, Sicherheitskontextbeschränkungen (SCCs) während der Installation von Trident Protect zu konfigurieren, wurde hinzugefügt.
- Unterstützung für die Konfiguration von nodeSelector während der Installation von Trident Protect hinzugefügt.
- Unterstützung für HTTP/HTTPS-Ausgangsproxy für AppVault-Objekte hinzugefügt.
- Erweitert ResourceFilter, um den Ausschluss von clusterweiten Ressourcen zu ermöglichen.
- Unterstützung für das AWS-Sitzungstoken in S3 AppVault Anmeldeinformationen hinzugefügt.
- Unterstützung für die Ressourcenerfassung nach der Ausführung von Pre-Snapshot-Hooks hinzugefügt.

Korrekturen

- Die Verwaltung temporärer Volumes wurde verbessert, um die ONTAP Volume-Wiederherstellungswarteschlange zu umgehen.
- SCC-Anmerkungen werden jetzt auf die ursprünglichen Werte zurückgesetzt.
- Verbesserte Wiederherstellungseffizienz mit Unterstützung für parallele Vorgänge.
- Erweiterte Unterstützung für Ausführungs-Hook-Timeouts für größere Anwendungen.

Änderungen in 24.10.1

Verbesserungen

- **Kubernetes**: Unterstützung für Kubernetes 1.32 hinzugefügt.

- iSCSI-Verbindungsstatus-Erkennung und Protokollierung hinzugefügt, wenn iSCSI-Sitzungen protokolliert sein sollten, es aber nicht sind ("Problem #961").

Korrekturen

- Fehlende Knoten-IP-Adressen aus automatischen Exportrichtlinien wurden korrigiert ("Problem #965").
- Automatisches Umschalten der Exportrichtlinien auf volumenbasierte Richtlinien für ONTAP-NAS-Economy wurde behoben.
- Die Abhängigkeiten von Trident und Trident-ASUP wurden aktualisiert, um CVE-2024-45337 und CVE-2024-45310 zu beheben.
- Abmeldungen für zeitweise fehlerhafte Nicht-CHAP-Portale während der iSCSI Selbstreparatur entfernt ("Problem #961").

Änderungen in 24.10

Verbesserungen

- Google Cloud NetApp Volumes-Treiber ist jetzt allgemein für NFS-Volumes verfügbar und unterstützt zonenbasierte Bereitstellung.
- GCP Workload Identity wird als Cloud Identity für Google Cloud NetApp Volumes mit GKE verwendet.
- Added `formatOptions` Konfigurationsparameter zu den ONTAP-SAN- und ONTAP-SAN-Economy-Treibern, um Benutzern zu ermöglichen, LUN-Formatoptionen anzugeben.
- Die Mindestvolumengröße für Azure NetApp Files wurde auf 50 GiB reduziert. Die neue Mindestgröße für Azure wird voraussichtlich im November allgemein verfügbar sein.
- Added `denyNewVolumePools` Konfigurationsparameter, um die Treiber ONTAP-NAS-Economy und ONTAP-SAN-Economy auf bereits vorhandene FlexVol-Pools zu beschränken.
- Hinzugefügte Erkennung für das Hinzufügen, die Entfernung oder das Umbenennen von Aggregaten aus dem SVM über alle ONTAP-Treiber hinweg.
- 18 MiB Overhead zu LUKS-LUNs hinzugefügt, um sicherzustellen, dass die gemeldete PVC-Größe nutzbar ist.
- Verbesserte Fehlerbehandlung beim Stagen und Unstagen von ONTAP-SAN- und ONTAP-SAN-Economy-Knoten, damit Unstage Geräte nach einem fehlgeschlagenen Stage entfernen kann.
- Es wurde ein benutzerdefinierter Rollengenerator hinzugefügt, der es Kunden ermöglicht, eine minimalistische Rolle für Trident in ONTAP zu erstellen.
- Zusätzliche Protokollierung zur Fehlerbehebung hinzugefügt `lsscsi` ("Problem #792").

Kubernetes

- Neue Trident-Funktionen für Kubernetes-native Workflows hinzugefügt:
  - Datenschutz
  - Datenmigration
  - Katastrophenwiederherstellung
  - Anwendungsmobilität

  "Erfahren Sie mehr über Trident Protect".

- Ein neues Flag `--k8s-api-qps` wurde den Installationsprogrammen hinzugefügt, um den von Trident für die Kommunikation mit dem Kubernetes API-Server verwendeten QPS-Wert festzulegen.
- Added `--node-prep` Flag zu den Installationsprogrammen für die automatische Verwaltung von Speicherprotokollabhängigkeiten auf Kubernetes-Clusterknoten. Kompatibilität mit Amazon Linux 2023 iSCSI-Speicherprotokoll getestet und verifiziert
- Unterstützung für das erzwungene Trennen von ONTAP-NAS-Economy-Volumes während Szenarien eines nicht ordnungsgemäßen Herunterfahrens von Knoten hinzugefügt.
- Neue ONTAP-NAS-Economy-NFS-Volumes verwenden Exportrichtlinien pro Qtree, wenn die `autoExportPolicy` Backend-Option verwendet wird. Qtrees werden nur zum Zeitpunkt der Veröffentlichung knotenbeschränkten Exportrichtlinien zugeordnet, um die Zugriffskontrolle und Sicherheit zu verbessern. Bestehende Qtrees werden auf das neue Exportrichtlinienmodell umgestellt, wenn Trident das Volume von allen Knoten entfernt, um dies ohne Beeinträchtigung aktiver Workloads zu tun.
- Unterstützung für Kubernetes 1.31 hinzugefügt.

Experimentelle Verbesserungen

- Tech Preview für Fibre Channel-Unterstützung im ONTAP-SAN-Treiber hinzugefügt.

Korrekturen

- **Kubernetes**:
  - Behobener Rancher admission webhook verhindert Trident Helm-Installationen ("Problem #839").
  - Fester Affinitätsschlüssel in Helm-Chart-Werten ("Problem #898").
  - Behoben tridentControllerPluginNodeSelector/tridentNodePluginNodeSelector funktioniert nicht mit dem Wert "true" ("Problem #899").
  - Gelöschte temporäre Snapshots, die während des Klonvorgangs erstellt wurden ("Problem #901").
- Unterstützung für Windows Server 2019 hinzugefügt.
- Behoben `go mod tidy` im Trident-Repository ("Problem #767").

Veraltete Funktionen

- **Kubernetes:**
  - Minimale unterstützte Kubernetes-Version auf 1.25 aktualisiert.
  - Unterstützung für POD Security Policy entfernt.

Produkt-Rebranding

Ab der Version 24.10 wird Astra Trident in Trident (NetApp Trident) umbenannt. Diese Umbenennung hat keine Auswirkungen auf Funktionen, unterstützte Plattformen oder die Interoperabilität von Trident.

Änderungen in 24.06

Verbesserungen

- **WICHTIG**: Der `limitVolumeSize` Parameter begrenzt nun die qtree-/LUN-Größen in den ONTAP economy-Treibern. Verwenden Sie den neuen `limitVolumePoolSize` Parameter, um die FlexVol-Größen in diesen Treibern zu steuern. ("Problem #341").

- Fähigkeit hinzugefügt, dass die iSCSI-Selbstreparatur SCSI-Scans anhand der exakten LUN-ID initiieren kann, wenn veraltete igroups verwendet werden ("Problem #883").
- Unterstützung für Volume-Klon- und Größenänderungsvorgänge wurde hinzugefügt, sodass diese auch dann erlaubt sind, wenn sich das Backend im Suspended-Modus befindet.
- Es wurde die Möglichkeit hinzugefügt, benutzerdefinierte Protokollierungseinstellungen für den Trident Controller an die Trident Node Pods weiterzugeben.
- Unterstützung in Trident hinzugefügt, um standardmäßig REST anstelle von ONTAPI (ZAPI) für ONTAP Versionen 9.15.1 und höher zu verwenden.
- Unterstützung für benutzerdefinierte Volumennamen und Metadaten auf den ONTAP-Speicher-Backends für neue persistente Volumen hinzugefügt.
- Erweiterte den `azure-netapp-files` (ANF)-Treiber, um das Snapshot-Verzeichnis standardmäßig automatisch zu aktivieren, wenn die NFS-Mount-Optionen so eingestellt sind, dass NFS Version 4.x verwendet wird.
- Unterstützung für Bottlerocket für NFS-Volumes hinzugefügt.
- Technische Vorschauunterstützung für Google Cloud NetApp Volumes hinzugefügt.

Kubernetes

- Unterstützung für Kubernetes 1.30 hinzugefügt.
- Funktion hinzugefügt, damit Trident DaemonSet beim Start Zombie-Mounts und verbleibende Tracking-Dateien bereinigen kann ("Problem #883").
- PVC-Annotation `trident.netapp.io/luksEncryption` für den dynamischen Import von LUKS-Volumes ("Problem #849").
- Dem ANF-Treiber wurde Topologiebewusstsein hinzugefügt.
- Unterstützung für Windows Server 2022-Knoten hinzugefügt.

Korrekturen

- Installationsfehler von Trident aufgrund veralteter Transaktionen wurden behoben.
- tridentctl wurde so korrigiert, dass Warnmeldungen von Kubernetes ("Problem #892") ignoriert werden.
- Die Priorität des Trident-Controllers `SecurityContextConstraint` wurde auf `0` ("Problem #887") geändert.
- ONTAP-Treiber akzeptieren jetzt Volume-Größen unter 20 MiB ("Ausgabe[#885").
- Trident wurde behoben, um zu verhindern, dass FlexVol-Volumes während der Größenänderung für den ONTAP-SAN-Treiber verkleinert werden.
- Behebung des Fehlers beim Import von ANF-Volume mit NFS v4.1.

Änderungen in 24.02

Verbesserungen

- Unterstützung für Cloud Identity hinzugefügt.
  - AKS mit ANF – Azure Workload Identity wird als Cloud identity verwendet.
  - EKS mit FSxN - AWS IAM role wird als Cloud-Identität verwendet.
- Unterstützung hinzugefügt, um Trident als Add-on auf einem EKS-Cluster über die EKS-Konsole zu

installieren.

- Fähigkeit hinzugefügt, die iSCSI Selbstreparatur zu konfigurieren und zu deaktivieren ("Problem #864").
- Amazon FSx-Persönlichkeit wurde den ONTAP-Treibern hinzugefügt, um die Integration mit AWS IAM und SecretsManager zu ermöglichen und damit Trident das Löschen von FSx-Volumes mit Backups ermöglichen kann ("Problem #453").

Kubernetes

- Unterstützung für Kubernetes 1.29 hinzugefügt.

Korrekturen

- Behobene ACP-Warnmeldungen, wenn ACP nicht aktiviert ist ("Problem #866").
- Vor der Durchführung einer Klon-Teilung während des Snapshot-Löschens bei ONTAP-Treibern wurde eine Verzögerung von 10 Sekunden hinzugefügt, wenn ein Klon mit dem Snapshot verknüpft ist.

Veraltete Funktionen

- Das in-toto-Attestierungsframework wurde aus den plattformübergreifenden Image-Manifesten entfernt.

Änderungen in 23.10

Korrekturen

- Feste Volumenerweiterung, wenn eine neu angeforderte Größe kleiner ist als die Gesamtvolumengröße für ontap-nas und ontap-nas-flexgroup Storage-Treiber ("Problem #834").
- Feste Volume-Größe, um beim Import für die ontap-nas- und ontap-nas-flexgroup-Speichertreiber nur die nutzbare Größe des Volumes anzuzeigen ("Problem #722").
- Die Namenskonvertierung von FlexVol für ONTAP-NAS-Economy wurde korrigiert.
- Problem mit der Trident-Initialisierung auf einem Windows-Knoten beim Neustart des Knotens behoben.

Verbesserungen

Kubernetes

Unterstützung für Kubernetes 1.28 hinzugefügt.

Trident

- Unterstützung für die Verwendung von Azure Managed Identities (AMI) mit dem azure-netapp-files Speichertreiber hinzugefügt.
- Unterstützung für NVMe over TCP für den ONTAP-SAN-Treiber hinzugefügt.
- Funktion hinzugefügt, um die Bereitstellung eines Volumes zu pausieren, wenn das Backend vom Benutzer in den angehaltenen Zustand versetzt wird ("Problem #558").

Änderungen in 23.07.1

Kubernetes: Problem mit der Daemonset-Löschung behoben, um Upgrades ohne Ausfallzeiten zu unterstützen ("Problem #740").

Änderungen in 23.07

Korrekturen

Kubernetes

- Trident-Upgrade wurde so korrigiert, dass alte Pods, die im Status „Terminating“ feststecken, ignoriert werden ("Problem #740").
- Zur Definition "transient-trident-version-pod" wurde eine Toleranz hinzugefügt ("Problem #795").

Trident

- ONTAPI (ZAPI)-Anfragen wurden behoben, um sicherzustellen, dass LUN-Seriennummern beim Abrufen von LUN-Attributen abgefragt werden, um Geister-iSCSI-Geräte während Node-Staging-Operationen zu identifizieren und zu beheben.
- Fehlerbehandlung im Speichertreibercode behoben ("Problem #816").
- Behobene Quotenanpassung bei Verwendung von ONTAP-Treibern mit use-rest=true.
- Die Erstellung von LUN-Klonen in ontap-san-economy wurde behoben.
- Das Feld „Veröffentlichungsinformationen“ wurde von `rawDevicePath` auf `devicePath` zurückgesetzt; es wurde Logik hinzugefügt, um das `devicePath` Feld (in einigen Fällen) zu füllen und wiederherzustellen.

Verbesserungen

Kubernetes

- Unterstützung für den Import von vorab bereitgestellten Snapshots hinzugefügt.
- Minimierte Bereitstellung und Daemonset-Linux-Berechtigungen ("Problem #817").

Trident

- Das Statusfeld für „online“-Volumes und Snapshots wird nicht mehr gemeldet.
- Aktualisiert den Backend-Status, wenn das ONTAP Backend offline ist ("Problem Nr. 801", "#543").
- Die LUN-Seriennummer wird immer während des ControllerVolumePublish-Workflows abgerufen und veröffentlicht.
- Zusätzliche Logik wurde hinzugefügt, um die Seriennummer und Größe des iSCSI-Multipath-Geräts zu überprüfen.
- Zusätzliche Überprüfung für iSCSI-Volumes, um sicherzustellen, dass das richtige Multipath-Gerät entstaged wird.

Experimentelle Verbesserung

Technische Vorschauunterstützung für NVMe über TCP für den ONTAP-SAN-Treiber hinzugefügt.

Dokumentation

Es wurden zahlreiche organisatorische und formattechnische Verbesserungen vorgenommen.

Veraltete Funktionen

Kubernetes

- Unterstützung für v1beta1 snapshots entfernt.
- Unterstützung für Pre-CSI-Volumes und Speicherklassen entfernt.
- Minimale unterstützte Kubernetes-Version auf 1.22 aktualisiert.

Änderungen in 23.04

> Das erzwungene Trennen von ONTAP-SAN-* Volumes wird nur von Kubernetes-Versionen unterstützt, bei denen das Feature „Nicht ordnungsgemäßes Herunterfahren von Knoten“ aktiviert ist. Das erzwungene Trennen muss während der Installation mithilfe des `--enable-force-detach` Trident-Installer-Flags aktiviert werden.

Korrekturen

- Trident Operator wurde so korrigiert, dass er bei Angabe in der Spezifikation IPv6 localhost für die Installation verwendet.
- Die Berechtigungen der Trident Operator-Clusterrolle wurden so angepasst, dass sie mit den Bundle-Berechtigungen übereinstimmen ("Problem #799").
- Problem beim Anhängen von Raw-Block-Volume auf mehreren Knoten im RWX-Modus behoben.
- Die Unterstützung für das Klonen von FlexGroup und den Volumenimport für SMB-Volumes wurde behoben.
- Problem behoben, bei dem der Trident Controller nicht sofort heruntergefahren werden konnte ("Problem #811").
- Es wurde ein Fix hinzugefügt, um alle Initiatorgruppen-Namen aufzulisten, die einer bestimmten LUN zugeordnet sind, die mit ontap-san-* Treibern bereitgestellt wurde.
- Ein Fix wurde hinzugefügt, um externen Prozessen zu ermöglichen, bis zum Abschluss ausgeführt zu werden.
- Kompilierungsfehler für die s390-Architektur behoben ("Problem #537").
- Falsche Protokollierungsebene während Volume-Mount-Operationen behoben ("Problem #781").
- Potenzieller Typzusicherungsfehler behoben ("Problem #802").

Verbesserungen

- Kubernetes:
  - Unterstützung für Kubernetes 1.27 hinzugefügt.
  - Unterstützung für den Import von LUKS-Volumes hinzugefügt.
  - Unterstützung für ReadWriteOncePod PVC-Zugriffsmodus hinzugefügt.
  - Unterstützung für das erzwungene Trennen von ONTAP-SAN-* Volumes während nicht ordnungsgemäßer Knotenabschaltungen hinzugefügt.
  - Alle ONTAP-SAN-* Volumes verwenden nun knotenspezifische Initiatorgruppen. LUNs werden nur dann Initiatorgruppen zugeordnet, wenn sie aktiv auf diesen Knoten veröffentlicht sind, um unsere Sicherheitslage zu verbessern. Bestehende Volumes werden opportunistisch auf das neue Initiatorgruppen-Schema umgestellt, sobald Trident feststellt, dass dies sicher ist und aktive Workloads

nicht beeinträchtigt werden ("Problem #758").

- Verbesserte Trident-Sicherheit durch Bereinigung nicht verwendeter, von Trident verwalteter Initiatorgruppen aus ONTAP-SAN-* Backends.

- Unterstützung für SMB-Volumes mit Amazon FSx zu den ontap-nas-economy- und ontap-nas-flexgroup-Speichertreibern hinzugefügt.
- Unterstützung für SMB-Freigaben mit den ontap-nas-, ontap-nas-economy- und ontap-nas-flexgroup-Speichertreibern hinzugefügt.
- Unterstützung für arm64-Knoten hinzugefügt ("Problem #732").
- Verbesserte Trident-Abschaltprozedur durch Deaktivierung der API-Server zuerst ("Problem #811").
- Plattformübergreifende Build-Unterstützung für Windows- und arm64-Hosts zum Makefile hinzugefügt; siehe BUILD.md.

Veraltete Funktionen

Kubernetes: Backend-bezogene igroups werden beim Konfigurieren der Treiber ontap-san und ontap-san-economy nicht mehr erstellt ("Problem #758").

Änderungen in 23.01.1

Korrekturen

- Trident Operator wurde so korrigiert, dass er bei Angabe in der Spezifikation IPv6 localhost für die Installation verwendet.
- Die Berechtigungen der Trident Operator-Clusterrolle wurden so angepasst, dass sie mit den Bundle-Berechtigungen übereinstimmen "Problem #799".
- Ein Fix wurde hinzugefügt, um externen Prozessen zu ermöglichen, bis zum Abschluss ausgeführt zu werden.
- Problem beim Anhängen von Raw-Block-Volume auf mehreren Knoten im RWX-Modus behoben.
- Die Unterstützung für das Klonen von FlexGroup und den Volumenimport für SMB-Volumes wurde behoben.

Änderungen in 23.01

Kubernetes 1.27 wird jetzt in Trident unterstützt. Bitte aktualisieren Sie Trident, bevor Sie Kubernetes aktualisieren.

Korrekturen

- Kubernetes: Es wurden Optionen hinzugefügt, um die Erstellung von Pod-Sicherheitsrichtlinien auszuschließen und so Trident-Installationen über Helm zu reparieren ("Probleme #783, #794").

Verbesserungen

Kubernetes

- Unterstützung für Kubernetes 1.26 hinzugefügt.
- Verbesserte allgemeine Trident RBAC Ressourcenauslastung ("Problem #757").
- Automatisierung hinzugefügt, um fehlerhafte oder veraltete iSCSI-Sitzungen auf Hostknoten zu erkennen

und zu beheben.

- Unterstützung für die Erweiterung von LUKS-verschlüsselten Volumes hinzugefügt.
- Kubernetes: Unterstützung für die Rotation von Anmeldeinformationen für LUKS-verschlüsselte Volumes hinzugefügt.

Trident

- Unterstützung für SMB-Volumes mit Amazon FSx for NetApp ONTAP wurde dem ontap-nas-Speichertreiber hinzugefügt.
- Unterstützung für NTFS-Berechtigungen bei Verwendung von SMB-Volumes hinzugefügt.
- Unterstützung für Speicherpools für GCP-Volumes mit CVS-Servicelevel hinzugefügt.
- Unterstützung für die optionale Verwendung von flexgroupAggregateList beim Erstellen von FlexGroups mit dem ontap-nas-flexgroup Storage-Treiber hinzugefügt.
- Verbesserte Leistung des ontap-nas-economy-Speichertreibers bei der Verwaltung mehrerer FlexVol-Volumes
- Aktivierte dataLIF-Updates für alle ONTAP NAS-Speichertreiber.
- Die Trident-Bereitstellungs- und DaemonSet-Namenskonvention wurde aktualisiert, um das Betriebssystem des Host-Knotens widerzuspiegeln.

Veraltete Funktionen

- Kubernetes: Mindestunterstützte Kubernetes-Version auf 1.21 aktualisiert.
- DataLIFs sollten bei der Konfiguration von `ontap-san` oder `ontap-san-economy` Treibern nicht mehr angegeben werden.

Änderungen in 22.10

Sie müssen die folgenden wichtigen Informationen lesen, bevor Sie auf Trident 22.10 aktualisieren.

Wichtige Informationen zu Trident 22.10

- Kubernetes 1.25 wird jetzt in Trident unterstützt. Sie müssen Trident auf 22.10 aktualisieren, bevor Sie auf Kubernetes 1.25 aktualisieren.
- Trident setzt nun die Verwendung der Multipathing-Konfiguration in SAN-Umgebungen strikt durch, mit einem empfohlenen Wert von `find_multipaths: no` in der multipath.conf-Datei.

  Die Verwendung einer Konfiguration ohne Multipathing oder die Verwendung von `find_multipaths: yes` oder `find_multipaths: smart` Wert in der Datei multipath.conf führt zu Mount-Fehlern. Trident hat die Verwendung von `find_multipaths: no` seit der Version 21.07 empfohlen.

Korrekturen

- Problem behoben, das spezifisch für ONTAP Backend war, das mit `credentials` Feld erstellt wurde und während des Upgrades auf 22.07.0 nicht online ging ("Problem #759").
- **Docker:** Es wurde ein Problem behoben, das dazu führte, dass das Docker-Volume-Plugin in einigen Umgebungen ("Problem #548" und "Problem #760") nicht gestartet werden konnte.
- SLM-Problem speziell für ONTAP SAN-Backends behoben, um sicherzustellen, dass nur eine Teilmenge

der dataLIFs, die zu den meldenden Knoten gehören, veröffentlicht wird.

- Leistungsproblem behoben, bei dem beim Anhängen eines Volumes unnötige Scans nach iSCSI-LUNs auftraten.
- Granulare Wiederholungsversuche innerhalb des Trident iSCSI-Workflows wurden entfernt, um ein schnelles Scheitern zu ermöglichen und die externen Wiederholungsintervalle zu reduzieren.
- Behobenes Problem, bei dem ein Fehler zurückgegeben wurde, wenn ein iSCSI-Gerät gespült wurde, während das entsprechende Multipath-Gerät bereits gespült worden war.

Verbesserungen

- Kubernetes:
  - Unterstützung für Kubernetes 1.25 hinzugefügt. Sie müssen Trident auf 22.10 aktualisieren, bevor Sie auf Kubernetes 1.25 aktualisieren.
  - Ein separates ServiceAccount, ClusterRole und ClusterRoleBinding wurde für die Trident-Bereitstellung und das DaemonSet hinzugefügt, um zukünftige Berechtigungserweiterungen zu ermöglichen.
  - Unterstützung für "Namensraumübergreifende Volumenfreigabe" hinzugefügt.
- Alle Trident `ontap-*` Speichertreiber funktionieren jetzt mit der ONTAP REST API.
- Neuer Operator-yaml (`bundle_post_1_25.yaml`) ohne ein `PodSecurityPolicy` hinzugefügt, um Kubernetes 1.25 zu unterstützen.
- Hinzugefügt "Unterstützung für LUKS-verschlüsselte Datenträger" für `ontap-san` und `ontap-san-economy` Speichertreiber.
- Unterstützung für Windows Server 2019-Knoten hinzugefügt.
- Hinzugefügt "Unterstützung für SMB-Volumes auf Windows-Knoten" durch den `azure-netapp-files` Speichertreiber.
- Automatische MetroCluster-Umschalterkennung für ONTAP-Treiber ist jetzt allgemein verfügbar.

Veraltete Funktionen

- **Kubernetes:** Mindestunterstützte Kubernetes-Version auf 1.20 aktualisiert.
- Astra Data Store (ADS) Treiber entfernt.
- Unterstützung für `yes` und `smart` Optionen für `find_multipaths` bei der Konfiguration von Worker-Node-Multipathing für iSCSI entfernt.

Änderungen in 22.07

Korrekturen

Kubernetes

- Problem bei der Verarbeitung von booleschen und numerischen Werten für den Knotenselektor bei der Konfiguration von Trident mit Helm oder dem Trident Operator behoben. ("GitHub Issue #700")
- Ein Problem bei der Fehlerbehandlung von Nicht-CHAP-Pfaden wurde behoben, sodass kubelet im Fehlerfall einen erneuten Versuch unternimmt. "GitHub Issue #736")

Verbesserungen

- Umstellung von k8s.gcr.io auf registry.k8s.io als Standard-Registry für CSI-Images
- ONTAP-SAN-Volumes verwenden ab sofort knotenspezifische Initiatorgruppen und ordnen LUNs nur dann Initiatorgruppen zu, wenn sie aktiv auf diesen Knoten veröffentlicht sind, um unsere Sicherheitslage zu verbessern. Bestehende Volumes werden opportunistisch auf das neue Initiatorgruppen-Schema umgestellt, sobald Trident feststellt, dass dies sicher ist und laufende Workloads nicht beeinträchtigt werden.
- Eine ResourceQuota ist jetzt bei Trident-Installationen enthalten, um sicherzustellen, dass Trident DaemonSet eingeplant wird, wenn der PriorityClass-Verbrauch standardmäßig begrenzt ist.
- Unterstützung für Netzwerkfunktionen zum Azure NetApp Files-Treiber hinzugefügt. ("GitHub Issue #717")
- Tech-Preview: Automatische MetroCluster-Umschaltungserkennung zu ONTAP-Treibern hinzugefügt. ("GitHub Issue #228")

Veraltete Funktionen

- **Kubernetes:** Mindestunterstützte Kubernetes-Version auf 1.19 aktualisiert.
- Die Backend-Konfiguration erlaubt nicht mehr mehrere Authentifizierungstypen in einer einzigen Konfiguration.

Entfernungen

- AWS CVS driver (veraltet seit 22.04) wurde entfernt.
- Kubernetes
  - Unnötige SYS_ADMIN-Berechtigung aus den Node-Pods entfernt.
  - Reduziert nodeprep auf einfache Hostinformationen und aktive Diensterkennung, um bestmöglich zu bestätigen, dass NFS/iSCSI-Services auf den Worker-Knoten verfügbar sind.

Dokumentation

Ein neuer "Pod-Sicherheitsstandards" (PSS)-Abschnitt wurde hinzugefügt, der die von Trident bei der Installation aktivierten Berechtigungen detailliert beschreibt.

Änderungen in 22.04

NetApp verbessert und erweitert seine Produkte und Dienstleistungen kontinuierlich. Hier sind einige der neuesten Funktionen in Trident. Informationen zu früheren Versionen finden Sie unter "Frühere Versionen der Dokumentation".

> Wenn Sie von einer früheren Trident-Version aktualisieren und Azure NetApp Files verwenden, ist der `location` Konfigurationsparameter nun ein obligatorisches Singleton-Feld.

Korrekturen

- Verbesserte Analyse von iSCSI-Initiatornamen. ("GitHub Issue #681")
- Problem behoben, bei dem CSI-Speicherklassenparameter nicht zulässig waren. ("GitHub Issue #598")
- Doppelte Schlüsseldeklaration in Trident CRD behoben. ("GitHub Issue #671")
- Fehlerhafte CSI-Snapshot-Protokolle korrigiert. ("GitHub Issue #629"))

- Problem beim Aufheben der Veröffentlichung von Volumes auf gelöschten Knoten behoben. ("GitHub Issue #691")
- Die Behandlung von Dateisysteminkonsistenzen auf Blockgeräten wurde hinzugefügt. ("GitHub Issue #656")
- Problem beim Abrufen von Auto-Support-Images beim Setzen des `imageRegistry`-Flags während der Installation behoben. ("GitHub Issue #715")
- Problem behoben, bei dem der Azure NetApp Files-Treiber ein Volume mit mehreren Exportregeln nicht klonen konnte.

Verbesserungen

- Eingehende Verbindungen zu den sicheren Endpunkten von Trident erfordern nun mindestens TLS 1.3. ("GitHub Issue #698")
- Trident fügt nun HSTS-Header zu den Antworten von seinen sicheren Endpunkten hinzu.
- Trident versucht nun, die Azure NetApp Files Unix-Berechtigungsfunktion automatisch zu aktivieren.
- **Kubernetes**: Trident Daemonset läuft jetzt mit der Prioritätsklasse „system-node-critical“. ("GitHub Issue #694")

Entfernungen

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

Änderungen in 22.01.1

Korrekturen

- Problem beim Aufheben der Veröffentlichung von Volumes auf gelöschten Knoten behoben. ("GitHub Issue #691")
- Panik beim Zugriff auf Nullfelder für Aggregatspeicherplatz in ONTAP-API-Antworten behoben.

Änderungen in 22.01.0

Korrekturen

- **Kubernetes:** Erhöhen Sie die Backoff-Wiederholungszeit für die Knotenregistrierung bei großen Clustern.
- Behobenes Problem, bei dem der azure-netapp-files-Treiber durch mehrere Ressourcen mit demselben Namen verwirrt werden konnte.
- ONTAP SAN IPv6 DataLIFs funktionieren jetzt, wenn sie mit Klammern angegeben werden.
- Problem behoben, bei dem der Versuch, ein bereits importiertes Volume zu importieren, EOF zurückgab und die PVC im Status „Ausstehend“ verblieb. ("GitHub Issue #489")
- Problem behoben, bei dem die Trident-Leistung nachließ, wenn mehr als 32 Snapshots auf einem SolidFire-Volume erstellt wurden.
- SHA-1 wurde bei der Erstellung von SSL-Zertifikaten durch SHA-256 ersetzt.
- Azure NetApp Files-Treiber wurde behoben, um doppelte Ressourcennamen zuzulassen und Vorgänge auf einen einzigen Standort zu beschränken.
- Azure NetApp Files-Treiber wurde behoben, um doppelte Ressourcennamen zuzulassen und Vorgänge auf einen einzigen Standort zu beschränken.

Verbesserungen

- Kubernetes-Erweiterungen:
  - Unterstützung für Kubernetes 1.23 hinzugefügt.
  - Fügen Sie Planungsoptionen für Trident Pods hinzu, wenn diese über Trident Operator oder Helm installiert werden. ("GitHub Issue #651")
- Regionsübergreifende Volumes im GCP-Treiber zulassen. ("GitHub Issue #633")
- Unterstützung für die Option „unixPermissions“ zu Azure NetApp Files-Volumes hinzugefügt. ("GitHub Issue #666")

Veraltete Funktionen

Trident REST interface kann nur an den Adressen 127.0.0.1 oder [::1] lauschen und Anfragen bedienen

Änderungen in 21.10.1

> Die Version v21.10.0 weist ein Problem auf, das den Trident Controller in einen CrashLoopBackOff-Zustand versetzen kann, wenn ein Node aus dem Kubernetes-Cluster entfernt und dann wieder hinzugefügt wird. Dieses Problem ist in v21.10.1 (GitHub issue 669) behoben.

Korrekturen

- Potenzielle Race Condition beim Importieren eines Volumes auf einem GCP CVS-Backend, die zu einem Fehler beim Import führte, wurde behoben.
- Es wurde ein Problem behoben, das dazu führen konnte, dass der Trident Controller in einen CrashLoopBackOff Zustand geriet, wenn ein Knoten aus dem Kubernetes-Cluster entfernt und anschließend wieder hinzugefügt wurde (GitHub issue 669).
- Problem behoben, bei dem SVMs nicht mehr gefunden wurden, wenn kein SVM-Name angegeben wurde (GitHub issue 612).

Änderungen in 21.10.0

Korrekturen

- Problem behoben, bei dem Klone von XFS-Volumes nicht auf demselben Knoten wie das Quellvolume eingebunden werden konnten (GitHub issue 514).
- Problem behoben, bei dem Trident beim Herunterfahren einen schwerwiegenden Fehler protokollierte (GitHub issue 597).
- Kubernetes-bezogene Korrekturen:
  - Geben Sie den belegten Speicherplatz eines Volumes als minimales restoreSize zurück, wenn Snapshots mit `ontap-nas` und `ontap-nas-flexgroup` Treibern erstellt werden (GitHub issue 645).
  - Problem behoben, bei dem `Failed to expand filesystem` ein Fehler nach einer Volumenänderung protokolliert wurde (GitHub issue 560).
  - Problem behoben, bei dem ein Pod im `Terminating` Zustand hängen bleiben konnte (GitHub issue 572).
  - Der Fall, in dem ein `ontap-san-economy` FlexVol möglicherweise mit Snapshot-LUNs gefüllt war,

wurde behoben (GitHub issue 533).

- Problem mit dem benutzerdefinierten YAML-Installer bei Verwendung eines anderen Images behoben (GitHub issue 613).
- Korrektur der Snapshot-Größenberechnung (GitHub issue 611).
- Problem behoben, bei dem alle Trident-Installationsprogramme normales Kubernetes als OpenShift (GitHub Issue 639) identifizieren konnten.
- Der Trident-Operator wurde so korrigiert, dass die Abgleichung gestoppt wird, wenn der Kubernetes-API-Server nicht erreichbar ist (GitHub issue 599).

Verbesserungen

- Unterstützung für `unixPermissions` Option für GCP-CVS Performance-Volumes hinzugefügt.
- Unterstützung für skalierungsoptimierte CVS-Volumes in GCP im Bereich von 600 GiB bis 1 TiB hinzugefügt.
- Kubernetes-bezogene Verbesserungen:
  - Unterstützung für Kubernetes 1.22 hinzugefügt.
  - Trident-Operator und Helm-Chart wurden für Kubernetes 1.22 aktiviert (GitHub issue 628).
  - Operator-Image zum `tridentctl` images-Befehl hinzugefügt (GitHub issue 570).

Experimentelle Erweiterungen

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

Bekannte Probleme

Bekannte Probleme identifizieren Schwierigkeiten, die eine erfolgreiche Nutzung des Produkts verhindern könnten.

- Beim Upgrade eines Kubernetes-Clusters von Version 1.24 auf 1.25 oder höher, auf dem Trident installiert ist, müssen Sie values.yaml aktualisieren, um `excludePodSecurityPolicy` auf `true` zu setzen oder `--set excludePodSecurityPolicy=true` zum `helm upgrade` Befehl hinzuzufügen, bevor Sie das Cluster aktualisieren können.
- Trident erzwingt nun ein leeres `fsType` (`fsType=""` für Volumes, die das `fsType` nicht in ihrer StorageClass angegeben haben. Bei der Arbeit mit Kubernetes 1.17 oder höher unterstützt Trident die Angabe eines leeren `fsType` für NFS-Volumes. Für iSCSI-Volumes müssen Sie das `fsType` in Ihrer StorageClass festlegen, wenn Sie ein `fsGroup` mithilfe eines Security Context erzwingen.
- Wenn ein Backend über mehrere Trident-Instanzen hinweg verwendet wird, sollte jede Backend-Konfigurationsdatei einen anderen `storagePrefix` Wert für ONTAP-Backends haben oder einen anderen `TenantName` für SolidFire-Backends verwenden. Trident kann Volumes, die von anderen Trident-Instanzen erstellt wurden, nicht erkennen. Der Versuch, ein vorhandenes Volume auf ONTAP- oder SolidFire-Backends zu erstellen, ist erfolgreich, da Trident die Volume-Erstellung als idempotente Operation behandelt. Wenn `storagePrefix` oder `TenantName` nicht unterschiedlich sind, kann es zu Namenskonflikten bei Volumes kommen, die auf demselben Backend erstellt werden.
- Bei der Installation von Trident (mithilfe von `tridentctl` oder dem Trident Operator) und der Verwendung von `tridentctl` zur Verwaltung von Trident sollten Sie sicherstellen, dass die `KUBECONFIG`

Umgebungsvariable gesetzt ist. Dies ist erforderlich, um den Kubernetes-Cluster anzugeben, gegen den `tridentctl` arbeiten soll. Wenn Sie mit mehreren Kubernetes-Umgebungen arbeiten, sollten Sie sicherstellen, dass die `KUBECONFIG` Datei korrekt eingebunden wird.

- Um die Online-Speicherplatzfreigabe für iSCSI-PVs durchzuführen, benötigt das zugrunde liegende Betriebssystem auf dem Worker-Knoten möglicherweise Mount-Optionen für das Volume. Dies gilt für RHEL/Red Hat Enterprise Linux CoreOS (RHCOS)-Instanzen, die die `discard` "Mount-Option" benötigen; stellen Sie sicher, dass die discard mountOption in Ihrer [`StorageClass` enthalten ist, um die Online-Blockverwerfung zu unterstützen.
- Wenn Sie mehr als eine Instanz von Trident pro Kubernetes-Cluster haben, kann Trident nicht mit anderen Instanzen kommunizieren und kann andere von ihnen erstellte Volumes nicht entdecken, was zu unerwartetem und fehlerhaftem Verhalten führt, wenn mehr als eine Instanz innerhalb eines Clusters ausgeführt wird. Es sollte nur eine Instanz von Trident pro Kubernetes-Cluster geben.
- Wenn Trident-basierte `StorageClass`Objekte aus Kubernetes gelöscht werden, während Trident offline ist, entfernt Trident die entsprechenden Speicherklassen nicht aus seiner Datenbank, wenn es wieder online ist. Sie sollten diese Speicherklassen mit `tridentctl`oder der REST-API löschen.
- Wenn ein Benutzer ein von Trident bereitgestelltes PV löscht, bevor er das zugehörige PVC löscht, wird das zugrunde liegende Volume von Trident nicht automatisch gelöscht. Sie sollten das Volume über `tridentctl` oder die REST-API entfernen.
- ONTAP kann nicht gleichzeitig mehr als eine FlexGroup bereitstellen, es sei denn, die Menge der Aggregate ist für jede Bereitstellungsanfrage eindeutig.
- Bei der Verwendung von Trident über IPv6 sollten Sie `managementLIF` und `dataLIF` in der Backend-Definition in eckigen Klammern angeben. Zum Beispiel `[fd20:8b1e:b258:2000:f816:3eff:feec:0]`.

> Sie können `dataLIF` auf einem ONTAP SAN-Backend nicht angeben. Trident erkennt alle verfügbaren iSCSI-LIFs und verwendet sie, um die Multipath-Sitzung herzustellen.

- Wenn Sie den `solidfire-san` Treiber mit OpenShift 4.5 verwenden, 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 verfügbar.

Weitere Informationen

- "Trident GitHub"
- "Trident Blogs"

Frühere Versionen der Dokumentation

Falls Sie nicht Trident 26.02 verwenden, ist die Dokumentation für frühere Versionen basierend auf der "Trident Support-Lebenszyklus" verfügbar.

- "Trident 25.10"
- "Trident 25.06"
- "Trident 25.02"
- "Trident 24.10"
- "Trident 24.06"

- "Trident 24.02"
- "Trident 23.10"
- "Trident 23.07"
- "Trident 23.04"

NetApp Trident-Unterstützung für ONTAP ASA r2-Speichersysteme

NetApp Trident 25.02 und höher unterstützt NetApp ASA r2-Systeme als Speicher-Backends. Weitere Informationen sind unter "ASA r2-Systeme" verfügbar.

ASA r2-Systeme benötigen den `ontap-san` Treiber. Trident unterstützt den `ontap-san-economy` Treiber für ASA r2-Systeme nicht.

Wenn Sie `ontap-san` als `storageDriverName` in der Backend-Konfiguration angeben, erkennt Trident das ASA r2-Speichersystem automatisch.

Trident bietet eingeschränkten Datenschutz für ASA r2-Systeme mit Trident protect.

Unterstützte SAN-Protokolle hängen von Ihrer Trident Version ab:

- Trident 25.02 und höher unterstützt iSCSI.
- Trident 25.06 und höher unterstützt NVMe/TCP zusätzlich zu iSCSI.
- Trident 25.10 und höher unterstützt neben iSCSI und NVMe/TCP auch FC.

Sie müssen der Storage Virtual Machine (SVM) für ONTAP-Backend-Speicher mindestens ein Aggregat zuweisen. Siehe "Zuweisung von Aggregaten zu SVMs in ASA r2 systems" für Anweisungen.

Unterstützte Operationen

- Bereitstellung persistenter Volumes (PVs)
- Dynamische Volumenbereitstellung
- Erstellen und Löschen von Volumes
- Klonen von Volumes
- Erweiterung von Volumes
- Verwaltung von Speicherklassen

Nicht unterstützte Operationen

- LUKS-Verschlüsselung
- SnapMirror Volume-Replikation
- Begrenzung der Aggregatnutzung
- Platzreservierungsmodus
- Snapshots
- Tiering

Weitere Informationen finden Sie unter "ONTAP SAN-Konfigurationsoptionen und Beispiele".

Bekannte Probleme

Bekannte Probleme beschreiben Fehler, die die erfolgreiche Nutzung dieser Produktversion verhindern könnten.

Die folgenden bekannten Probleme betreffen die aktuelle Version:

VolumeSnapshots erreicht nicht den ReadyToUse-Zustand

Es gibt ein bekanntes Problem mit dem `csi-snapshotter` Sidecar. In allen Kubernetes-Versionen verhindert VolumeGroupSnapshots `v1beta1`, dass VolumeSnapshots den `ReadyToUse` Zustand erreichen.

Zwei Workarounds sind verfügbar:

1. Löschen Sie die VolumeGroupSnapshots CRDs, um VolumeGroupSnapshots zu deaktivieren, und installieren Sie dann Trident neu.
2. Installieren Sie VolumeGroupSnapshots `v1beta2` und `snapshot-controller` Version 8.4.0 oder höher und installieren Sie dann Trident erneut. VolumeGroupSnapshots funktionieren auf Kubernetes-Versionen vor v1.34 nicht.

Das Wiederherstellen von Restic-Backups großer Dateien kann fehlschlagen

Beim Wiederherstellen von Dateien ab 30 GB aus einem Amazon S3-Backup, das mit Restic erstellt wurde, kann der Wiederherstellungsvorgang fehlschlagen. Als Workaround sichern Sie die Daten mit Kopia als Daten-Mover (Kopia ist der Standard-Daten-Mover für Backups). Siehe "Schützen Sie Anwendungen mit Trident Protect" für Anweisungen.

Los geht’s

Erfahren Sie mehr über Trident

Erfahren Sie mehr über Trident

Trident ist ein vollständig unterstütztes Open-Source-Projekt, das von NetApp gepflegt wird. Es wurde entwickelt, um Ihnen dabei zu helfen, die Persistenzanforderungen Ihrer containerisierten Anwendung mithilfe von branchenüblichen Schnittstellen wie dem Container Storage Interface (CSI) zu erfüllen.

Was ist Trident?

Netapp Trident ermöglicht die Nutzung und Verwaltung von Speicherressourcen über alle gängigen NetApp Speicherplattformen hinweg, in der Public Cloud oder On-Premises, einschließlich On-Premises ONTAP-Clustern (AFF, FAS und ASA), ONTAP Select, Cloud Volumes ONTAP, Element Software (NetApp HCI, SolidFire), Azure NetApp Files, Amazon FSx for NetApp ONTAP und Google Cloud NetApp Volumes.

Trident ist ein Container Storage Interface (CSI) konformer dynamischer Storage-Orchestrator, der sich nativ mit "Kubernetes" integriert. Trident läuft als einzelner Controller-Pod plus ein Node-Pod auf jedem Worker-Knoten im Cluster. Siehe "Trident-Architektur" für Details.

Trident bietet zudem eine direkte Integration mit dem Docker-Ökosystem für NetApp Speicherplattformen. Das NetApp Docker Volume Plugin (nDVP) unterstützt die Bereitstellung und Verwaltung von Speicherressourcen von der Speicherplattform zu Docker-Hosts. Siehe "Trident für Docker bereitstellen" für Details.

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

Unterstützte Kubernetes-Plattformen

Trident wird auf einer Reihe von Kubernetes-Distributionen und Plattformen unterstützt.

Unterstützte Plattformen sind: * Upstream Kubernetes * Red Hat OpenShift * SUSE Harvester 1.7.0 (ONTAP iSCSI)

Kubernetes-Integration mit NetApp Produkten

Das NetApp Portfolio an Speicherprodukten integriert sich in viele Aspekte eines Kubernetes-Clusters und bietet fortschrittliche Datenverwaltungsfunktionen, die die Funktionalität, Fähigkeit, Performance und Verfügbarkeit der Kubernetes-Bereitstellung verbessern.

Amazon FSx for NetApp ONTAP

"Amazon FSx for NetApp ONTAP" ist ein vollständig verwalteter AWS-Service, mit dem Sie Dateisysteme starten und ausführen können, die vom NetApp ONTAP Storage-Betriebssystem betrieben werden.

Azure NetApp Files

"Azure NetApp Files" ist ein Azure-Dateifreigabedienst der Enterprise-Klasse, der von NetApp betrieben wird. Sie können Ihre anspruchsvollsten dateibasierten Workloads nativ in Azure ausführen, mit der Leistung und dem umfassenden Datenmanagement, das Sie von NetApp erwarten.

Cloud Volumes ONTAP

"Cloud Volumes ONTAP" ist eine rein softwarebasierte Storage-Appliance, die die ONTAP Datenmanagement-Software in der Cloud ausführt.

Google Cloud NetApp Volumes

"Google Cloud NetApp Volumes" ist ein vollständig verwalteter Dateispeicherdienst in Google Cloud, der hochleistungsfähigen Dateispeicher der Enterprise-Klasse bereitstellt.

Element Software

"Element" ermöglicht es Speicheradministratoren, Workloads zu konsolidieren, indem Leistung garantiert und ein vereinfachter sowie optimierter Speicher-Footprint ermöglicht wird.

NetApp HCI

"NetApp HCI" vereinfacht die Verwaltung und Skalierung des Rechenzentrums durch die Automatisierung von Routineaufgaben und ermöglicht Infrastrukturadministratoren, sich auf wichtigere Funktionen zu konzentrieren.

Trident kann Speichergeräte für containerisierte Anwendungen direkt auf der zugrunde liegenden NetApp HCI-Speicherplattform bereitstellen und verwalten.

NetApp ONTAP

"NetApp ONTAP" ist das NetApp Multiprotokoll-Betriebssystem mit einheitlichem Speicher, das fortschrittliche Datenverwaltungsfunktionen für jede Anwendung bietet.

ONTAP-Systeme verfügen über All-Flash-, Hybrid- oder All-HDD-Konfigurationen und bieten zahlreiche Bereitstellungsmodelle: On-Premises FAS-, AFA- und ASA-Cluster, ONTAP Select und Cloud Volumes ONTAP. Trident unterstützt diese ONTAP-Bereitstellungsmodelle.

Trident-Architektur

Trident läuft als einzelner Controller-Pod plus ein Node-Pod auf jedem Worker-Knoten im Cluster. Der Node-Pod muss auf jedem Host ausgeführt werden, auf dem Sie möglicherweise ein Trident Volume einbinden möchten.

Controller-Pods und Node-Pods verstehen

Trident wird als einzelner Trident Controller Pod und einer oder mehrere Trident Node Pods auf dem Kubernetes-Cluster bereitgestellt und verwendet standardmäßige Kubernetes , um die Bereitstellung von CSI-Plugins zu vereinfachen. "Kubernetes CSI Sidecar Container" werden von der Kubernetes Storage Community gewartet.

Kubernetes "Knotenselektoren" und "Tolerations und Taints" werden verwendet, um einen Pod auf einen bestimmten oder bevorzugten Knoten zu beschränken. Sie können Knotenselektoren und Toleranzen für Controller- und Knoten-Pods während der Trident-Installation konfigurieren.

- Das Controller-Plugin übernimmt die Bereitstellung und Verwaltung von Volumes, wie Snapshots und Größenänderungen.
- Das Node-Plugin übernimmt das Anbinden des Speichers an den Knoten.

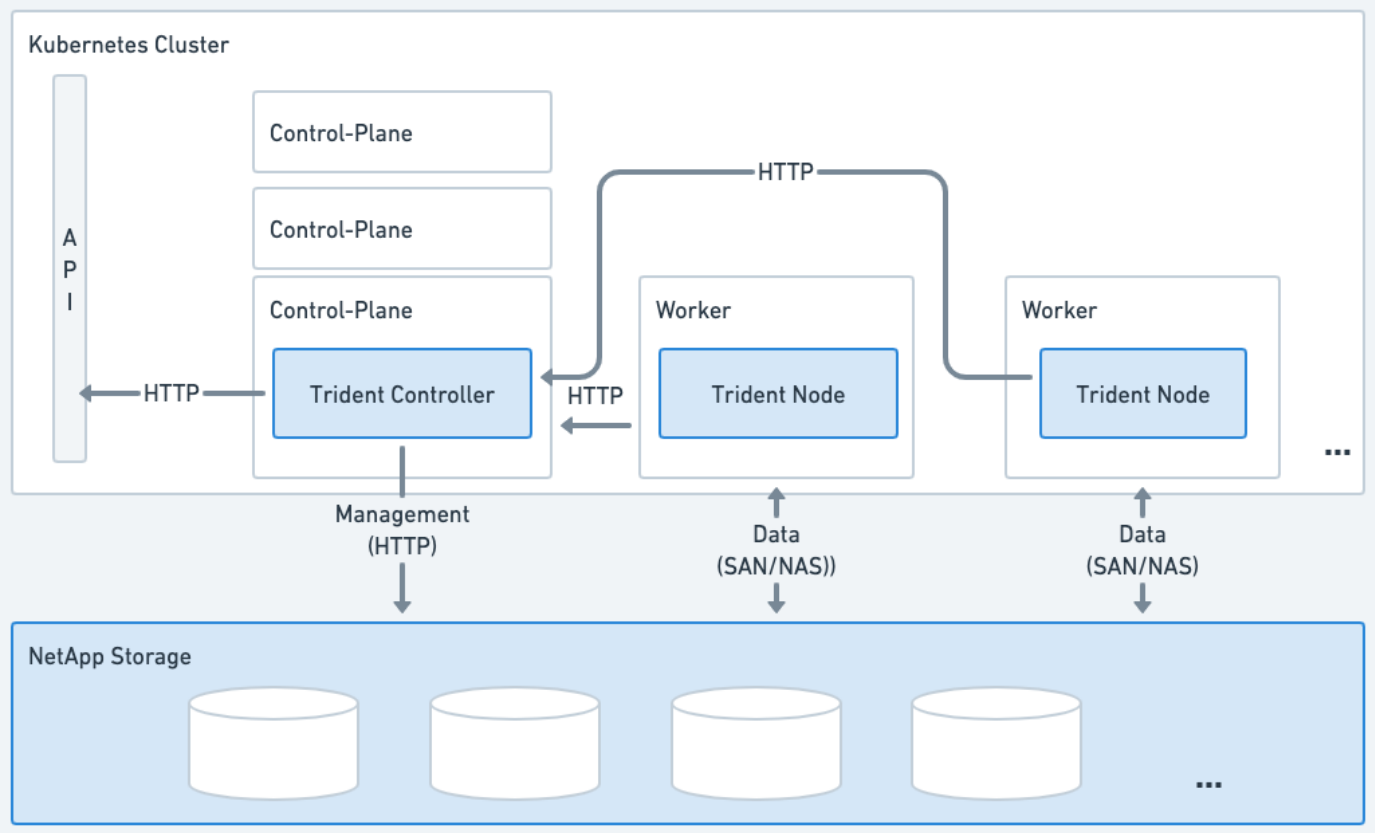


Abbildung 1. Trident wurde auf dem Kubernetes-Cluster bereitgestellt

Trident Controller Pod

Der Trident Controller Pod ist ein einzelner Pod, auf dem das CSI Controller Plugin ausgeführt wird.

- Verantwortlich für die Bereitstellung und Verwaltung von Volumes im NetApp Storage
- Verwaltet durch eine Kubernetes Deployment
- Kann je nach Installationsparametern auf den Steuerungsebene- oder Arbeitsknoten ausgeführt werden.

Abbildung 2. Trident Controller Pod-Diagramm

Trident Node Pods

Trident Node Pods sind privilegierte Pods, auf denen das CSI Node Plugin ausgeführt wird.

- Verantwortlich für das Ein- und Aushängen von Storage für Pods, die auf dem Host ausgeführt werden
- Verwaltet von einem Kubernetes DaemonSet
- Muss auf jedem Knoten ausgeführt werden, der NetApp Storage einbinden wird

Abbildung 3. Trident Node Pod Diagramm

Unterstützte Kubernetes-Clusterarchitekturen

Trident wird mit den folgenden Kubernetes-Architekturen unterstützt:

Kubernetes-Clusterarchitekturen	Unterstützt	Standardinstallation
Einzelner Master, Compute	Ja	Ja
Mehrere Master, Compute	Ja	Ja
Meister, `etcd`, berechnen	Ja	Ja
Master, Infrastruktur, Compute	Ja	Ja

Konzepte

Bereitstellung

Die Bereitstellung in Trident hat zwei Hauptphasen. Die erste Phase ordnet eine Speicherklasse dem Satz geeigneter Backend-Speicherpools zu und erfolgt als notwendige Vorbereitung vor der Bereitstellung. Die zweite Phase umfasst die eigentliche Volume-Erstellung und erfordert die Auswahl eines Speicherpools aus denen, die der Speicherklasse des ausstehenden Volumes zugeordnet sind.

Speicherklassenzuordnung

Die Zuordnung von Backend-Speicherpools zu einer Speicherklasse basiert sowohl auf den angeforderten Attributen der Speicherklasse als auch auf deren `storagePools`, `additionalStoragePools` und `excludeStoragePools` Listen. Wenn Sie eine Speicherklasse erstellen, vergleicht Trident die von jedem seiner Backends angebotenen Attribute und Pools mit denen, die von der Speicherklasse angefordert werden. Wenn die Attribute und der Name eines Speicherpools mit allen angeforderten Attributen und Poolnamen übereinstimmen, fügt Trident diesen Speicherpool der Menge geeigneter Speicherpools für diese Speicherklasse hinzu. Zusätzlich fügt Trident alle in der `additionalStoragePools` Liste aufgeführten Speicherpools dieser Menge hinzu, selbst wenn deren Attribute nicht alle oder keine der angeforderten Attribute der Speicherklasse erfüllen. Sie sollten die `excludeStoragePools` Liste verwenden, um Speicherpools für eine Speicherklasse außer Kraft zu setzen und aus der Nutzung zu entfernen. Trident führt einen ähnlichen Prozess jedes Mal durch, wenn Sie ein neues Backend hinzufügen, prüft, ob dessen Speicherpools die Anforderungen der bestehenden Speicherklassen erfüllen, und entfernt alle, die als ausgeschlossen markiert wurden.

Volumenerstellung

Trident verwendet dann die Zuordnungen zwischen Speicherklassen und Speicherpools, um zu bestimmen, wo Volumes bereitgestellt werden. Wenn Sie ein Volume erstellen, ermittelt Trident zunächst die Speicherpools für die Speicherklasse dieses Volumes, und wenn Sie ein Protokoll für das Volume angeben, entfernt Trident diejenigen Speicherpools, die das angeforderte Protokoll nicht bereitstellen können (zum Beispiel kann ein NetApp HCI/SolidFire Backend kein dateibasiertes Volume bereitstellen, während ein ONTAP NAS Backend kein blockbasiertes Volume bereitstellen kann). Trident randomisiert die Reihenfolge dieser resultierenden Menge, um eine gleichmäßige Verteilung der Volumes zu ermöglichen, und iteriert dann durch sie, wobei versucht wird, das Volume nacheinander auf jedem Speicherpool bereitzustellen. Wenn dies bei einem gelingt, wird erfolgreich zurückgegeben und alle während des Prozesses aufgetretenen Fehler werden protokolliert. Trident gibt nur dann einen Fehler zurück, **wenn** die Bereitstellung auf **allen** für die angeforderte Speicherklasse und das Protokoll verfügbaren Speicherpools fehlschlägt.

Volume Snapshots

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

Erfahren Sie mehr über die Erstellung von Volume-Snapshots

- Für die `ontap-nas`, `ontap-san` und `azure-netapp-files` Treiber wird jedes Persistent Volume (PV) einem FlexVol volume zugeordnet. Dadurch werden Volume-Snapshots als NetApp Snapshots erstellt. Die NetApp Snapshot-Technologie bietet mehr Stabilität, Skalierbarkeit, Wiederherstellbarkeit und Leistung als konkurrierende Snapshot-Technologien. Diese Snapshot-Kopien sind äußerst effizient, sowohl in Bezug auf die benötigte Zeit für ihre Erstellung als auch auf den Speicherplatz.
- Für den `ontap-nas-flexgroup`-Treiber wird jedes Persistent Volume (PV) einem FlexGroup zugeordnet. Dadurch werden Volume-Snapshots als NetApp FlexGroup-Snapshots erstellt. Die NetApp Snapshot-Technologie bietet mehr Stabilität, Skalierbarkeit, Wiederherstellbarkeit und Leistung als konkurrierende Snapshot-Technologien. Diese Snapshot-Kopien sind äußerst effizient, sowohl in Bezug auf die benötigte Zeit für ihre Erstellung als auch auf den Speicherplatz.
- Für den `ontap-san-economy` Treiber werden PVs auf LUNs abgebildet, die auf gemeinsam genutzten FlexVol Volumes erstellt wurden. VolumeSnapshots von PVs werden erreicht, indem FlexClones der zugehörigen LUN durchgeführt werden. ONTAP FlexClone Technologie ermöglicht es, Kopien selbst der größten Datensätze nahezu augenblicklich zu erstellen. Kopien teilen sich Datenblöcke mit ihren übergeordneten Elementen und belegen keinen Speicherplatz außer dem, der für Metadaten benötigt wird.
- Für den `solidfire-san` Treiber wird jedes PV einer LUN zugeordnet, die auf der NetApp Element Software/NetApp HCI-Cluster erstellt wurde. VolumeSnapshots werden durch Element-Snapshots der zugrunde liegenden LUN repräsentiert. Diese Snapshots sind zeitpunktgenaue Kopien und beanspruchen nur wenig Systemressourcen und Speicherplatz.
- Bei der Arbeit mit dem `ontap-nas` und `ontap-san` Treiber sind ONTAP-Snapshots zeitpunktgenaue Kopien des FlexVol und belegen Speicherplatz auf dem FlexVol selbst. Dies kann dazu führen, dass der beschreibbare Speicherplatz im Volume mit der Zeit abnimmt, wenn Snapshots erstellt oder geplant werden. Eine einfache Möglichkeit, dies zu beheben, besteht darin, das Volume durch Resize über Kubernetes zu vergrößern. Eine weitere Option ist das Löschen von Snapshots, die nicht mehr benötigt werden. Wenn ein durch Kubernetes erstellter VolumeSnapshot gelöscht wird, löscht Trident den zugehörigen ONTAP-Snapshot. ONTAP-Snapshots, die nicht durch Kubernetes erstellt wurden, können ebenfalls gelöscht werden.

Mit Trident können Sie VolumeSnapshots verwenden, um neue PVs daraus zu erstellen. Das Erstellen von PVs aus diesen Snapshots erfolgt mithilfe der FlexClone-Technologie für unterstützte ONTAP-Backends. Beim Erstellen eines PVs aus einem Snapshot ist das zugrunde liegende Volume ein FlexClone des übergeordneten

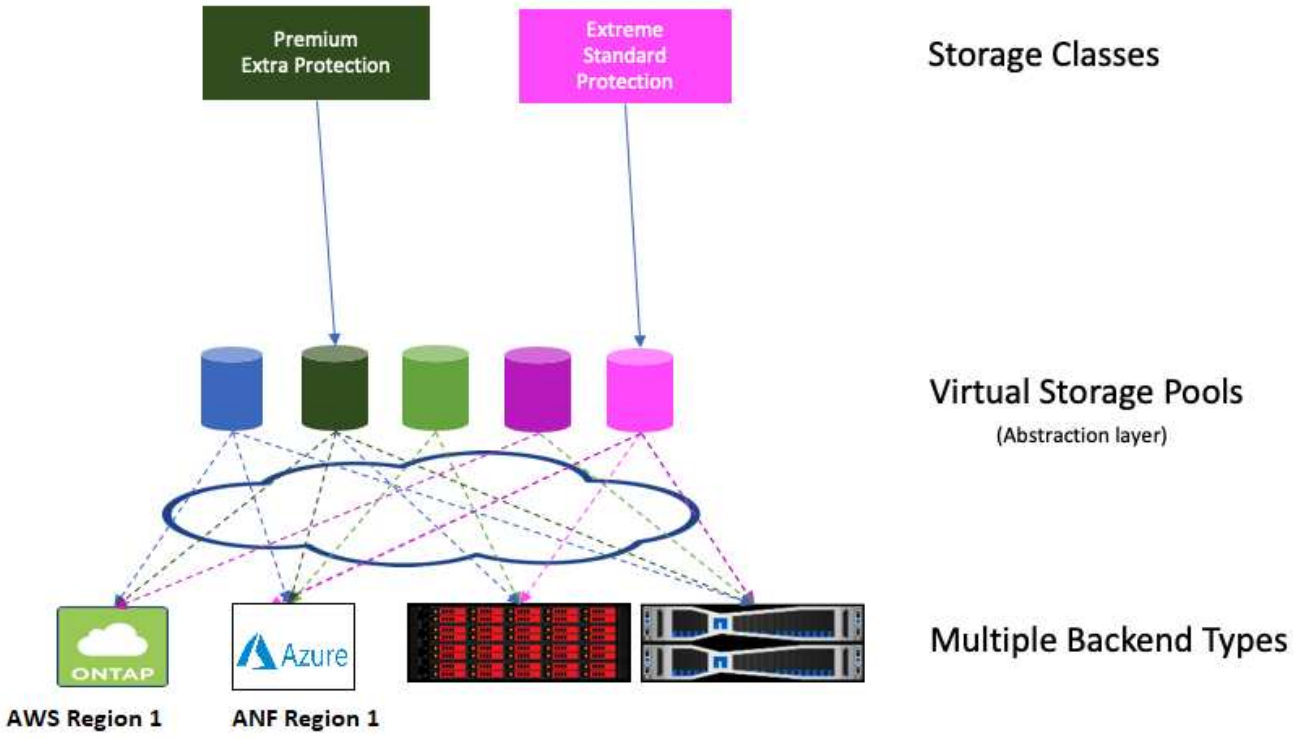
Volumes des Snapshots. Der `solidfire-san`Treiber verwendet Element-Software-Volume-Klone, um PVs aus Snapshots zu erstellen. Hier wird ein Klon des Element-Snapshots erstellt.

Virtuelle Pools

Virtuelle Pools bieten eine Abstraktionsschicht zwischen Trident-Speicher-Backends und Kubernetes `StorageClasses`. Sie ermöglichen es einem Administrator, Aspekte wie Standort, Leistung und Schutz für jedes Backend auf eine einheitliche, backendunabhängige Weise zu definieren, ohne eine `StorageClass` angeben zu müssen, welches physische Backend, welcher Backend-Pool oder welcher Backend-Typ verwendet werden soll, um die gewünschten Kriterien zu erfüllen.

Erfahren Sie mehr über virtuelle Pools

Der Speicheradministrator kann virtuelle Pools auf jedem der Trident Backends in einer JSON- oder YAML-Definitionsdatei definieren.



Alle Aspekte, die außerhalb der Liste der virtuellen Pools angegeben werden, sind global für das Backend und gelten für alle virtuellen Pools, während jeder virtuelle Pool einen oder mehrere Aspekte individuell angeben kann (wodurch globale Aspekte des Backends überschrieben werden).

- Beim Definieren virtueller Pools sollten Sie nicht versuchen, die Reihenfolge vorhandener virtueller Pools in einer Backend-Definition zu ändern.
- Wir raten davon ab, Attribute eines bestehenden virtuellen Pools zu ändern. Sie sollten einen neuen virtuellen Pool definieren, um Änderungen vorzunehmen.

Die meisten Aspekte werden in Backend-spezifischen Begriffen spezifiziert. Entscheidend ist, dass die Aspektwerte nicht außerhalb des Backend-Treibers zugänglich sind und nicht für den Abgleich in `StorageClasses` verfügbar sind. Stattdessen definiert der Administrator für jeden virtuellen Pool ein oder mehrere Labels. Jedes Label ist ein Schlüssel:Wert-Paar, und Labels können für verschiedene Backends identisch sein. Wie Aspekte können auch Labels pro Pool oder global für das Backend definiert werden. Im Gegensatz zu Aspekten, die vordefinierte Namen und Werte haben, kann der Administrator die Label-Schlüssel und -Werte nach Bedarf frei festlegen. Zur Vereinfachung können Speicheradministratoren Bezeichnungen pro virtuellem Pool definieren und Volumes nach Bezeichnung gruppieren.

Die virtuellen Poolbezeichnungen können mit diesen Zeichen definiert werden:

- Großbuchstaben `A-Z`
- Kleinbuchstaben `a-z`
- Zahlen `0-9`
- Unterstriche _
- Bindestriche -

A `StorageClass` identifiziert den zu verwendenden virtuellen Pool durch Verweis auf die Bezeichnungen innerhalb eines Selektorparameters. Selektoren für virtuelle Pools unterstützen die folgenden Operatoren:

Operator	Beispiel	Der Labelwert eines Pools muss:
`=`	performance=premium	Übereinstimmen
`!=`	performance!=extreme	Keine Übereinstimmung
`in`	Standort in (east, west)	Teil der Wertemenge sein
`notin`	performance notin (Silber, Bronze)	Nicht in der Wertemenge enthalten sein
`<key>`	Schutz	Existiert mit beliebigem Wert
`!<key>`	!Schutz	Nicht vorhanden

Volumenzugriffsgruppen

Erfahren Sie mehr darüber, wie Trident "Volumenzugriffsgruppen" verwendet.

> Ignorieren Sie diesen Abschnitt, wenn Sie CHAP verwenden, was empfohlen wird, um die Verwaltung zu vereinfachen und die unten beschriebene Skalierungsbeschränkung zu vermeiden. Wenn Sie Trident im CSI-Modus verwenden, können Sie diesen Abschnitt ebenfalls ignorieren. Trident verwendet CHAP, wenn es als erweiterter CSI-Provisioner installiert ist.

Erfahren Sie mehr über Volumenzugriffsgruppen

Trident kann Volume-Zugriffsgruppen verwenden, um den Zugriff auf die Volumes zu steuern, die es bereitstellt. Wenn CHAP deaktiviert ist, erwartet es, eine Zugriffsgruppe namens `trident` zu finden, es sei denn, Sie geben eine oder mehrere Zugriffsgruppen-IDs in der Konfiguration an.

Trident ordnet zwar neue Volumes den konfigurierten Zugriffsgruppen zu, erstellt oder verwaltet die Zugriffsgruppen jedoch nicht selbst. Die Zugriffsgruppen müssen existieren, bevor das Speicher-Backend zu Trident hinzugefügt wird, und sie müssen die iSCSI-IQNs von jedem Knoten im Kubernetes-Cluster enthalten, der potenziell die von diesem Backend bereitgestellten Volumes einbinden könnte. In den meisten

Installationen umfasst dies alle Worker-Knoten im Cluster.

Für Kubernetes-Cluster mit mehr als 64 Knoten sollten Sie mehrere Zugriffsgruppen verwenden. Jede Zugriffsgruppe kann bis zu 64 IQNs enthalten, und jedes Volume kann zu vier Zugriffsgruppen gehören. Mit den maximal vier konfigurierten Zugriffsgruppen kann jeder Knoten in einem Cluster mit bis zu 256 Knoten auf jedes Volume zugreifen. Die aktuellen Beschränkungen für Volume-Zugriffsgruppen finden Sie unter "hier,".

Wenn Sie die Konfiguration von einer, die die Standard `trident` Zugriffsgruppe verwendet, auf eine, die auch andere verwendet, ändern, fügen Sie die ID der `trident` Zugriffsgruppe in die Liste ein.

Schnellstart für Trident

Sie können Trident installieren und in wenigen Schritten mit der Verwaltung von Speicherressourcen beginnen. Bevor Sie beginnen, lesen Sie "Trident-Anforderungen".

Für Docker siehe "Trident für Docker".

1 Bereiten Sie den Worker-Knoten vor

Alle Worker-Knoten im Kubernetes-Cluster müssen die für Ihre Pods bereitgestellten Volumes einbinden können.

"Bereiten Sie den Worker-Knoten vor"

2 Installieren Sie Trident

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

"Trident installieren"

3 Erstelle ein Backend

Ein Backend definiert die Beziehung zwischen Trident und einem Speichersystem. Es gibt Trident vor, wie Trident mit diesem Speichersystem kommuniziert und wie Trident Volumes daraus bereitstellen soll.

"Ein Backend konfigurieren" for Ihr Speichersystem

4 Erstellen Sie eine Kubernetes-StorageClass

Das Kubernetes StorageClass-Objekt legt Trident als Provisioner fest und ermöglicht es Ihnen, eine Speicherklasse zu erstellen, um Volumes mit anpassbaren Attributen bereitzustellen. Trident erstellt eine passende Speicherklasse für Kubernetes-Objekte, die den Trident-Provisioner angeben.

"Erstellen Sie eine Speicherklasse"

5 Volume bereitstellen

Ein PersistentVolume (PV) ist eine physische Speicherressource, die vom Clusteradministrator auf einem

Kubernetes-Cluster bereitgestellt wird. Der PersistentVolumeClaim (PVC) ist eine Anfrage für den Zugriff auf das PersistentVolume im Cluster.

Erstellen Sie ein PersistentVolume (PV) und ein PersistentVolumeClaim (PVC), das die konfigurierte Kubernetes-StorageClass verwendet, um Zugriff auf das PV anzufordern. Anschließend können Sie das PV in einen Pod einbinden.

"Ein Volume bereitstellen"

Was kommt als Nächstes?

Sie können nun zusätzliche Backends hinzufügen, Speicherklassen verwalten, Backends verwalten und Volume-Operationen durchführen.

Anforderungen

Vor der Installation von Trident sollten Sie diese allgemeinen Systemvoraussetzungen überprüfen. Bestimmte Backends können zusätzliche Anforderungen haben.

Wichtige Informationen zu Trident

Sie müssen die folgenden wichtigen Informationen über Trident lesen.

Wichtige Informationen zu Trident

- Kubernetes 1.36 wird jetzt in Trident unterstützt. Trident sollte vor Kubernetes aktualisiert werden.
- Trident setzt die Verwendung der Multipathing-Konfiguration in SAN-Umgebungen strikt durch, mit einem empfohlenen Wert von `find_multipaths: no` in der multipath.conf-Datei.

  Die Verwendung einer Konfiguration ohne Multipathing oder die Verwendung von `find_multipaths: yes` oder `find_multipaths: smart` Wert in der Datei multipath.conf führt zu Mount-Fehlern. Trident hat die Verwendung von `find_multipaths: no` seit der Version 21.07 empfohlen.

Unterstützte Frontends (Orchestratoren)

Trident unterstützt mehrere Container-Engines und Orchestratoren, einschließlich der folgenden:

- Anthos On-Prem (VMware) und Anthos on bare metal 1.16
- Kubernetes 1.27 - 1.36
- OpenShift 4.12, 4.14 - 4.21 (Wenn Sie die iSCSI-Knotenvorbereitung mit OpenShift 4.19 verwenden möchten, ist die minimal unterstützte Trident-Version 25.06.1.)

Trident unterstützt weiterhin ältere OpenShift-Versionen in Übereinstimmung mit dem "Red Hat Extended Update Support (EUS) Release-Lebenszyklus", selbst wenn sie auf Kubernetes-Versionen basieren, die upstream nicht mehr offiziell unterstützt werden. Bei der Installation von Trident in solchen Fällen können Sie Warnmeldungen zur Kubernetes-Version getrost ignorieren.

- Rancher Kubernetes Engine 2 (RKE2) v1.28.x - 1.36.x

Trident arbeitet außerdem mit einer Vielzahl anderer vollständig verwalteter und selbstverwalteter Kubernetes-Angebote zusammen, einschließlich Google Kubernetes Engine (GKE), Amazon Elastic Kubernetes Services (EKS), Azure Kubernetes Service (AKS), Mirantis Kubernetes Engine (MKE) und VMWare Tanzu Portfolio.

Trident und ONTAP können als Speicheranbieter für "KubeVirt" verwendet werden.

Bevor Sie einen Kubernetes-Cluster von Version 1.25 auf 1.26 oder höher aktualisieren, auf dem Trident installiert ist, beachten Sie "Aktualisieren einer Helm-Installation".

Unterstützte Backends (Storage)

Um Trident zu verwenden, benötigen Sie eines oder mehrere der folgenden unterstützten Backends:

- Amazon FSx for NetApp ONTAP
- Azure NetApp Files
- Cloud Volumes ONTAP
- Google Cloud NetApp Volumes
- NetApp All SAN Array (ASA)
- Lokale FAS, AFF oder ASA r2 (iSCSI, NVMe/TCP und FC) mit ONTAP-Versionen unter NetApp voller oder eingeschränkter Unterstützung. Siehe "Software-Versions-Support".
- NetApp HCI/Element-Software 11 oder höher

Trident Unterstützung für KubeVirt und OpenShift Virtualisierung

Unterstützte Speichertreiber:

Trident unterstützt die folgenden ONTAP-Treiber für KubeVirt und OpenShift Virtualisierung:

- ontap-nas
- ontap-san (iSCSI, FCP, NVMe over TCP)
- ontap-san-economy (nur iSCSI)

Zu beachtende Punkte:

- Aktualisieren Sie die Speicherklasse, sodass der `fsType` Parameter (zum Beispiel: `fsType: "ext4"`) in der OpenShift Virtualisierungsumgebung vorhanden ist. Falls erforderlich, legen Sie den Volume-Modus explizit auf Block fest, indem Sie den `volumeMode=Block` Parameter in der `dataVolumeTemplates` verwenden, um CDI anzuweisen, Blockdaten-Volumes zu erstellen.
- *RWX-Zugriffsmodus für Block-Speichertreiber*: ontap-san (iSCSI, NVMe/TCP, FC) und ontap-san-economy (iSCSI) Treiber werden nur mit "volumeMode: Block" (Rohgerät) unterstützt. Für diese Treiber kann der `fstype` Parameter nicht verwendet werden, da die Volumes im Rohgerätemodus bereitgestellt werden.

- Für Live-Migrations-Workflows, bei denen der RWX-Zugriffsmodus erforderlich ist, sind diese Kombinationen unterstützt:
  - NFS + `volumeMode=Filesystem`
  - iSCSI + `volumeMode=Block` (Rohgerät)
  - NVMe/TCP + `volumeMode=Block` (Rohgerät)
  - FC + `volumeMode=Block` (Rohgerät)

Funktionsanforderungen

Die folgende Tabelle fasst die in dieser Version von Trident verfügbaren Funktionen und die unterstützten Kubernetes-Versionen zusammen.

Funktion	Kubernetes-Version	Feature gates erforderlich?
Trident	1.27 - 1.36	Nein
Volume-Snapshots	1.27 - 1.36	Nein
PVC aus Volume Snapshots	1.27 - 1.36	Nein
iSCSI-PV-Größenänderung	1.27 - 1.36	Nein
ONTAP Bidirektionales CHAP	1.27 - 1.36	Nein
Dynamische Exportrichtlinien	1.27 - 1.36	Nein
Trident Operator	1.27 - 1.36	Nein
CSI-Topologie	1.27 - 1.36	Nein

Getestete Host-Betriebssysteme

Obwohl Trident keine spezifischen Betriebssysteme offiziell unterstützt, sind die folgenden dafür bekannt, zu funktionieren:

- Von OpenShift Container Platform auf AMD64 und ARM64 unterstützte Red Hat Enterprise Linux CoreOS (RHCOS)-Versionen
- Red Hat Enterprise Linux (RHEL) 8 oder höher auf AMD64 und ARM64

  NVMe/TCP erfordert RHEL 9 oder später.

- Ubuntu 22.04 LTS oder höher auf AMD64 und ARM64
- Windows Server 2022
- SUSE Linux Enterprise Server (SLES) 15 oder höher

Standardmäßig läuft Trident in einem Container und ist daher auf jedem Linux-Worker lauffähig. Diese Worker

müssen jedoch in der Lage sein, die von Trident bereitgestellten Volumes mithilfe des Standard-NFS-Clients oder iSCSI-Initiators einzubinden, abhängig von den verwendeten Backends.

Das `tridentctl` Dienstprogramm läuft auch auf allen diesen Linux-Distributionen.

Hostkonfiguration

Alle Worker-Knoten im Kubernetes-Cluster müssen die für Ihre Pods bereitgestellten Volumes einbinden können. Um die Worker-Knoten vorzubereiten, müssen Sie je nach Treiberauswahl NFS-, iSCSI- oder NVMe-Tools installieren.

"Bereiten Sie den Worker-Knoten vor"

Speichersystemkonfiguration

Trident erfordert möglicherweise Änderungen an einem Speichersystem, bevor eine Backend-Konfiguration es verwenden kann.

"Backends konfigurieren"

Trident-Ports

Trident benötigt Zugriff auf bestimmte Ports für die Kommunikation.

"Trident-Ports"

Container-Images und zugehörige Kubernetes-Versionen

Für Installationen ohne Internetverbindung (Air-Gap) ist die folgende Liste eine Referenz der Container-Images, die zur Installation von Trident benötigt werden. Verwenden Sie den `tridentctl images`-Befehl, um die Liste der benötigten Container-Images zu überprüfen.

Für Trident 26.02 werden Container-Images benötigt.

Kubernetes-Versionen	Containerbild
v1.27.0, v1.28.0, v1.29.0, v1.30.0, v1.31.0, v1.32.0, v1.33.0, v1.34.0, v1.36.0	• docker.io/netapp/trident:26.02.0 • docker.io/netapp/trident-autosupport:26.02 • registry.k8s.io/sig-storage/csi-provisioner:v6.1.0 • registry.k8s.io/sig-storage/csi-attacher:v4.10.0 • registry.k8s.io/sig-storage/csi-resizer:v2.0.0 • registry.k8s.io/sig-storage/csi-snapshotter:v8.5.0 • registry.k8s.io/sig-storage/csi-node-driver-registrar:v2.15.0 • docker.io/netapp/trident-operator:26.02.0 (optional)

Trident installieren

Erfahren Sie mehr über die Trident-Installation

Um sicherzustellen, dass Trident in einer Vielzahl von Umgebungen und Organisationen installiert werden kann, bietet NetApp mehrere Installationsoptionen an. Sie können Trident mithilfe des Trident-Operators (manuell oder mit Helm) oder mit `tridentctl` installieren. Dieses Thema enthält wichtige Informationen zur Auswahl des richtigen Installationsprozesses für Sie.

Wichtige Informationen zu Trident 26.02

Sie müssen die folgenden wichtigen Informationen über Trident lesen.

Wichtige Informationen zu Trident

- Kubernetes 1.35 wird nun in Trident unterstützt. Trident sollte vor Kubernetes aktualisiert werden.
- Trident setzt die Verwendung der Multipathing-Konfiguration in SAN-Umgebungen strikt durch, mit einem empfohlenen Wert von `find_multipaths: no` in der multipath.conf-Datei.

  Die Verwendung einer Konfiguration ohne Multipathing oder die Verwendung von `find_multipaths: yes` oder `find_multipaths: smart` Wert in der Datei multipath.conf führt zu Mount-Fehlern. Trident hat die Verwendung von `find_multipaths: no` seit der Version 21.07 empfohlen.

Bevor Sie beginnen

Unabhängig vom Installationsweg müssen Sie Folgendes haben:

- Vollständige Berechtigungen für einen unterstützten Kubernetes-Cluster, auf dem eine unterstützte Kubernetes-Version ausgeführt wird und die erforderlichen Funktionen aktiviert sind. Überprüfen Sie die "Anforderungen" für Details.
- Zugriff auf ein unterstütztes NetApp Speichersystem.
- Fähigkeit, Volumes von allen Kubernetes-Worker-Knoten einzubinden.
- Ein Linux-Host mit `kubectl` (oder `oc`, wenn Sie OpenShift verwenden) installiert und konfiguriert, um den Kubernetes-Cluster zu verwalten, den Sie verwenden möchten.
- Die `KUBECONFIG` Umgebungsvariable, die auf Ihre Kubernetes-Clusterkonfiguration verweist.
- Wenn Sie Kubernetes mit Docker Enterprise verwenden "Folgen Sie deren Schritten, um den CLI-Zugriff zu aktivieren".
- Der Cluster muss privilegierte Workloads unterstützen.

Falls Sie sich noch nicht mit dem "Grundkonzepte" vertraut gemacht haben, ist jetzt ein guter Zeitpunkt dafür.

Wählen Sie Ihre Installationsmethode

Wählen Sie die für Sie passende Installationsmethode. Sie sollten auch die Überlegungen zu "Wechsel zwischen Methoden" prüfen, bevor Sie Ihre Entscheidung treffen.

Verwendung des Trident-Operators

Ob manuell oder mit Helm – der Trident-Operator ist eine großartige Möglichkeit, die Installation zu vereinfachen und Trident-Ressourcen dynamisch zu verwalten. Sie können sogar "Passen Sie Ihre Trident-Operator-Bereitstellung an" unter Verwendung der Attribute in der `TridentOrchestrator` Custom Resource (CR).

Zu den Vorteilen der Verwendung des Trident Operators gehören:

Trident Objekterstellung

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

- ServiceAccount für den Operator
- ClusterRole und ClusterRoleBinding zum ServiceAccount
- Dedizierte PodSecurityPolicy (für Kubernetes 1.25 und früher)
- Der Operator selbst

Ressourcenverantwortung

Der clusterweite Trident-Operator verwaltet Ressourcen einer Trident-Installation auf Clusterebene. Dies mindert Fehler, die bei der Verwaltung clusterweiter Ressourcen mit einem namensraumweiten Operator auftreten können. Dies ist für Selbstreparatur und das Einspielen von Patches unerlässlich.

Selbstreparatur-Fähigkeit

Der Operator überwacht die Trident-Installation und ergreift aktiv Maßnahmen, um Probleme zu beheben, beispielsweise wenn die Bereitstellung gelöscht oder versehentlich geändert wird. Ein `trident-operator-<generated-id>` Pod wird erstellt, der eine `TridentOrchestrator` CR mit einer Trident-Installation verknüpft. Dies stellt sicher, dass sich nur eine Instanz von Trident im Cluster befindet und deren Einrichtung kontrolliert wird, sodass die Installation idempotent ist. Wenn Änderungen an der Installation vorgenommen werden (wie das Löschen der Bereitstellung oder des Node-Daemonsets), erkennt der Operator diese und behebt sie einzeln.

Einfache Aktualisierungen bestehender Installationen

Sie können eine bestehende Bereitstellung mithilfe des Operators problemlos aktualisieren. Sie müssen lediglich die `TridentOrchestrator` CR bearbeiten, um Aktualisierungen an einer Installation vorzunehmen.

Nehmen Sie zum Beispiel ein Szenario, in dem Sie Trident so konfigurieren müssen, dass Debug-Protokolle generiert werden. Dazu patchen Sie Ihre `TridentOrchestrator` so, dass `spec.debug` auf `true` gesetzt wird:

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

Nachdem `TridentOrchestrator` aktualisiert wurde, verarbeitet der Operator die Updates und patcht die bestehende Installation. Dies kann die Erstellung neuer Pods auslösen, um die Installation entsprechend anzupassen.

Saubere Neuinstallation

Der clusterweite Trident-Operator ermöglicht die saubere Entfernung clusterweiter Ressourcen. Benutzer können Trident vollständig deinstallieren und problemlos neu installieren.

Automatische Kubernetes-Upgrade-Verarbeitung

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

> Wird der Cluster auf eine nicht unterstützte Version aktualisiert, verhindert der Operator die Installation von Trident. Falls Trident bereits mit dem Operator installiert wurde, wird eine Warnung angezeigt, die darauf hinweist, dass Trident auf einer nicht unterstützten Kubernetes-Version installiert ist.

Verwendung `tridentctl`

Wenn Sie eine bestehende Installation haben, die aktualisiert werden muss, oder wenn Sie Ihre Installation stark anpassen möchten, sollten Sie in Betracht ziehen. Dies ist die herkömmliche Methode zur Bereitstellung von Trident.

Sie können verwenden, um die Manifeste für Trident-Ressourcen zu generieren. Dies umfasst die Bereitstellung, das DaemonSet, das Dienstkonto und die Clusterrolle, die Trident im Rahmen der Installation erstellt.

Ab Version 22.04 werden AES-Schlüssel nicht mehr bei jeder Trident-Installation neu generiert. Mit dieser Version installiert Trident ein neues geheimes Objekt, das über zukünftige Installationen hinweg erhalten bleibt. Das bedeutet, `tridentctl` dass unter Version 22.04 frühere Versionen von Trident deinstalliert werden können, ältere Versionen jedoch nicht Installationen der Version 22.04 deinstallieren können. Wählen Sie die passende Installationsmethode.

Wählen Sie Ihren Installationsmodus

Bestimmen Sie Ihren Bereitstellungsprozess basierend auf dem von Ihrer Organisation benötigten *Installationsmodus* (Standard, Offline oder Remote).

Standardinstallation

Dies ist die einfachste Methode, Trident zu installieren, und funktioniert in den meisten Umgebungen, die keine Netzwerkbeschränkungen haben. Der Standardinstallationsmodus verwendet Standardregistries, um die erforderlichen Trident (`docker.io` und CSI (`registry.k8s.io` Images zu speichern.

Wenn Sie den Standardmodus verwenden, führt das Trident-Installationsprogramm Folgendes aus:

- Ruft die Container-Images über das Internet ab
- Erstellt ein Deployment oder einen Node-Daemonset, der Trident Pods auf allen geeigneten Nodes im Kubernetes-Cluster startet

Offline-Installation

Der Offline-Installationsmodus kann in einer vom Netzwerk getrennten oder gesicherten Umgebung erforderlich sein. In diesem Szenario können Sie eine einzelne private, gespiegelte Registry oder zwei gespiegelte Registrys erstellen, um die benötigten Trident- und CSI-Images zu speichern.

Unabhängig von Ihrer Registry-Konfiguration müssen CSI-Images in einer einzigen Registry vorhanden sein.

Ferninstallation

Hier ist ein Überblick auf hoher Ebene über den Ablauf der Ferninstallation:

- Installieren Sie die entsprechende Version von `kubectl` auf dem Remote-Rechner, von dem aus Sie Trident bereitstellen möchten.
- Kopieren Sie die Konfigurationsdateien aus dem Kubernetes-Cluster und setzen Sie die `KUBECONFIG` Umgebungsvariable auf dem Remote-Rechner.
- Initiieren Sie einen `kubectl get nodes`-Befehl, um zu überprüfen, ob Sie eine Verbindung zum erforderlichen Kubernetes-Cluster herstellen können.
- Schließen Sie die Bereitstellung von der Remote-Maschine aus mit den Standardinstallationsschritten ab.

Wählen Sie den Prozess basierend auf Ihrer Methode und Ihrem Modus aus

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

Verfahren	Installationsmodus
Trident operator (manuell)	"Standardinstallation" "Offline-Installation"
Trident-Operator (Helm)	"Standardinstallation" "Offline-Installation"
`tridentctl`	"Standard- oder Offline-Installation"

Wechsel zwischen Installationsmethoden

Sie können Ihre Installationsmethode ändern. Beachten Sie vorher Folgendes:

- Verwenden Sie immer die gleiche Methode für die Installation und Deinstallation von Trident. Wenn Sie mit `tridentctl` bereitgestellt haben, sollten Sie die entsprechende Version der `tridentctl`-Binary verwenden, um Trident zu deinstallieren. Ebenso sollten Sie, wenn Sie mit dem Operator bereitstellen, die `TridentOrchestrator` CR bearbeiten und `spec.uninstall=true` setzen, um Trident zu deinstallieren.
- Wenn Sie eine operatorbasierte Bereitstellung haben, die Sie entfernen und stattdessen `tridentctl` zur Bereitstellung von Trident verwenden möchten, sollten Sie zuerst `TridentOrchestrator` bearbeiten und `spec.uninstall=true` setzen, um Trident zu deinstallieren. Löschen Sie dann `TridentOrchestrator` und die Operator-Bereitstellung. Sie können dann mit `tridentctl` installieren.
- Wenn Sie eine manuelle, operatorbasierte Bereitstellung haben und auf die Helm-basierte Trident Operator-Bereitstellung umsteigen möchten, sollten Sie den Operator zuerst manuell deinstallieren und dann die Helm-Installation durchführen. Dadurch kann Helm den Trident Operator mit den erforderlichen Labels und Annotationen bereitstellen. Wenn Sie dies nicht tun, schlägt Ihre Helm-basierte Trident Operator-Bereitstellung mit einem Label-Validierungsfehler und einem Annotations-Validierungsfehler fehl.
- Wenn Sie eine `tridentctl`-basierte Bereitstellung haben, können Sie eine Helm-basierte oder Operator-basierte Bereitstellung durchführen, ohne Trident zu deinstallieren.

Weitere bekannte Konfigurationsoptionen

Bei der Installation von Trident auf VMWare Tanzu Portfolio-Produkten:

- Das `--kubelet-dir`-Flag sollte auf den Speicherort des kubelet-Verzeichnisses gesetzt werden. Standardmäßig ist dies `/var/vcap/data/kubelet`.

  Die Angabe des kubelet-Speicherorts mit `--kubelet-dir` ist bekannt dafür, bei Trident Operator-, Helm- und `tridentctl`-Deployments zu funktionieren.

Installation mit dem Trident operator

Manuelle Bereitstellung des Trident Operators (Standardmodus)

Sie können den Trident-Operator manuell bereitstellen, um Trident zu installieren. Dieser

Prozess gilt für Installationen, bei denen die von Trident benötigten Container-Images nicht in einer privaten Registry gespeichert sind. Wenn Sie eine private Image-Registry haben, verwenden Sie die "Prozess für die Offline-Bereitstellung".

Wichtige Informationen zu Trident 26.02

Sie müssen die folgenden wichtigen Informationen über Trident lesen.

Wichtige Informationen zu Trident

- Kubernetes 1.35 wird nun in Trident unterstützt. Trident sollte vor Kubernetes aktualisiert werden.
- Trident setzt die Verwendung der Multipathing-Konfiguration in SAN-Umgebungen strikt durch, mit einem empfohlenen Wert von `find_multipaths: no` in der multipath.conf-Datei.

  Die Verwendung einer Konfiguration ohne Multipathing oder die Verwendung von `find_multipaths: yes` oder `find_multipaths: smart` Wert in der Datei multipath.conf führt zu Mount-Fehlern. Trident hat die Verwendung von `find_multipaths: no` seit der Version 21.07 empfohlen.

Den Trident-Operator manuell bereitstellen und Trident installieren

Überprüfen Sie "die Installationsübersicht", um sicherzustellen, dass Sie die Installationsvoraussetzungen erfüllt und die richtige Installationsoption für Ihre Umgebung ausgewählt haben.

Bevor Sie beginnen

Bevor Sie mit der Installation beginnen, melden Sie sich am Linux-Host an und vergewissern Sie sich, dass dieser ein funktionierendes "unterstützter Kubernetes-Cluster" verwaltet und dass Sie über die erforderlichen Berechtigungen verfügen.

Mit OpenShift verwenden Sie `oc` statt `kubectl` in allen folgenden Beispielen und melden Sie sich zuerst als **system:admin** an, indem Sie `oc login -u system:admin` oder `oc login -u kube-admin` ausführen.

1. Überprüfen Sie Ihre Kubernetes-Version:

```
kubectl version
```

2. Überprüfen Sie die Cluster-Administratorrechte:

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

3. Überprüfen Sie, ob Sie einen Pod starten können, der ein Image von Docker Hub verwendet, und Ihr Speichersystem über das Pod-Netzwerk erreichen können.

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

Schritt 1: Laden Sie das Trident-Installationspaket herunter

Das Trident-Installationspaket enthält alles, was Sie zum Bereitstellen des Trident-Operators und zum Installieren von Trident benötigen. Laden Sie die neueste Version des Trident-Installationsprogramms von "der Abschnitt *Assets* auf GitHub" herunter und extrahieren Sie sie.

```
wget https://github.com/NetApp/trident/releases/download/v26.02.0/trident-
installer-26.02.0.tar.gz
tar -xf trident-installer-26.02.0.tar.gz
cd trident-installer
```

Schritt 2: Erstellen Sie die `TridentOrchestrator` CRD

Erstellen Sie die `TridentOrchestrator` benutzerdefinierte Ressourcendefinition (CRD). Sie werden später eine `TridentOrchestrator` benutzerdefinierte Ressource erstellen. Verwenden Sie die entsprechende CRD-YAML-Version in `deploy/crds`, um die `TridentOrchestrator` CRD zu erstellen.

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

Schritt 3: Den Trident Operator einsetzen

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

- Für Cluster, auf denen Kubernetes 1.24 läuft, verwenden Sie `bundle_pre_1_25.yaml`.

- Für Cluster, auf denen Kubernetes 1.25 oder höher läuft, verwenden Sie `bundle_post_1_25.yaml`.

Bevor Sie beginnen

- Standardmäßig installiert das Trident-Installationsprogramm den Operator im `trident` Namespace. Falls der `trident` Namespace nicht existiert, erstellen Sie ihn mit:

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

- Um den Operator in einem anderen Namespace als dem `trident` Namespace bereitzustellen, aktualisieren Sie `serviceaccount.yaml`, `clusterrolebinding.yaml` und `operator.yaml` und generieren Sie Ihre Bundle-Datei mit dem `kustomization.yaml`.

  a. Erstellen Sie das `kustomization.yaml` mit dem folgenden Befehl, wobei *<bundle.yaml>* `bundle_pre_1_25.yaml` oder `bundle_post_1_25.yaml` abhängig von Ihrer Kubernetes-Version ist.

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

  b. Kompilieren Sie das Bundle mit dem folgenden Befehl, wobei *<bundle.yaml>* `bundle_pre_1_25.yaml` oder `bundle_post_1_25.yaml` entsprechend Ihrer Kubernetes-Version ist.

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

Schritte

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

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

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

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

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

Schritt 4: Erstellen Sie das `TridentOrchestrator` und installieren Sie Trident

Sie können jetzt die `TridentOrchestrator` erstellen und Trident installieren. Optional können Sie "Passen Sie Ihre Trident Installation individuell an" mit den Attributen in der `TridentOrchestrator`-Spezifikation arbeiten.

```
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
  nodePrep:
  - iscsi
Status:
  Current Installation Params:
    IPv6:                      false
    Autosupport Hostname:
    Autosupport Image:         netapp/trident-autosupport:26.02
    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:26.02.0
  Message:                  Trident installed  Namespace:
trident
  Status:                   Installed
  Version:                  v26.02.0
Events:
    Type Reason Age From Message ---- ------ ---- ---- -------Normal
    Installing 74s trident-operator.netapp.io Installing Trident Normal
    Installed 67s trident-operator.netapp.io Trident installed
```

Überprüfen Sie die Installation

Es gibt mehrere Möglichkeiten, Ihre Installation zu überprüfen.

Status mit `TridentOrchestrator` verwenden

Der Status von `TridentOrchestrator` zeigt an, ob die Installation erfolgreich war und zeigt die installierte Version von Trident an. Während der Installation ändert sich der Status von `TridentOrchestrator` `Installing` zu `Installed`. Wenn Sie den `Failed` Status beobachten und der Operator das Problem nicht selbst beheben kann, "Überprüfen Sie die Protokolle".

Status	Beschreibung
Installation	Der Operator installiert Trident mit diesem `TridentOrchestrator` CR.
Installiert	Trident wurde erfolgreich installiert.
Deinstallation	Der Betreiber deinstalliert Trident, weil `spec.uninstall=true`.
Deinstalliert	Trident ist deinstalliert.
Fehlgeschlagen	Der Operator konnte Trident nicht installieren, patchen, aktualisieren oder deinstallieren; der Operator wird automatisch versuchen, sich von diesem Zustand zu erholen. Wenn dieser Zustand weiterhin besteht, ist eine Fehlerbehebung erforderlich.
Aktualisierung	Der Betreiber aktualisiert eine bestehende Installation.
Fehler	Das `TridentOrchestrator` wird nicht verwendet. Ein anderer existiert bereits.

Verwendung des Pod-Erstellungsstatus

Sie können bestätigen, ob die Trident Installation abgeschlossen wurde, indem Sie den Status der erstellten Pods überprüfen:

```
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
```

Verwendung `tridentctl`

Sie können `tridentctl` verwenden, um die installierte Version von Trident zu überprüfen.

```
./tridentctl -n trident version

+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 26.02.0        | 26.02.0        |
+----------------+----------------+
```

Manuelle Bereitstellung des Trident Operators (Offline-Modus)

Sie können den Trident-Operator manuell bereitstellen, um Trident zu installieren. Dieser Vorgang gilt für Installationen, bei denen die von Trident benötigten Container-Images in einer privaten Registry gespeichert sind. Wenn Sie keine private Image-Registry besitzen, verwenden Sie die "Prozess für die Standardbereitstellung".

Wichtige Informationen zu Trident

Sie müssen die folgenden wichtigen Informationen über Trident lesen.

Wichtige Informationen zu Trident

- Kubernetes 1.35 wird nun in Trident unterstützt. Trident sollte vor Kubernetes aktualisiert werden.
- Trident setzt die Verwendung der Multipathing-Konfiguration in SAN-Umgebungen strikt durch, mit einem empfohlenen Wert von `find_multipaths: no` in der multipath.conf-Datei.

  Die Verwendung einer Konfiguration ohne Multipathing oder die Verwendung von `find_multipaths: yes` oder `find_multipaths: smart` Wert in der Datei multipath.conf führt zu Mount-Fehlern. Trident hat die Verwendung von `find_multipaths: no` seit der Version 21.07 empfohlen.

Den Trident-Operator manuell bereitstellen und Trident installieren

Überprüfen Sie "die Installationsübersicht", um sicherzustellen, dass Sie die Installationsvoraussetzungen erfüllt und die richtige Installationsoption für Ihre Umgebung ausgewählt haben.

Bevor Sie beginnen

Melden Sie sich am Linux-Host an und vergewissern Sie sich, dass er ein funktionierendes und "unterstützter Kubernetes-Cluster" verwaltet und dass Sie über die erforderlichen Berechtigungen verfügen.

Mit OpenShift verwenden Sie `oc` statt `kubectl` in allen folgenden Beispielen und melden Sie sich zuerst als **system:admin** an, indem Sie `oc login -u system:admin` oder `oc login -u kube-admin` ausführen.

1. Überprüfen Sie Ihre Kubernetes-Version:

```
kubectl version
```

2. Überprüfen Sie die Cluster-Administratorrechte:

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

3. Überprüfen Sie, ob Sie einen Pod starten können, der ein Image von Docker Hub verwendet, und Ihr Speichersystem über das Pod-Netzwerk erreichen können.

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

Schritt 1: Laden Sie das Trident-Installationspaket herunter

Das Trident-Installationspaket enthält alles, was Sie zum Bereitstellen des Trident-Operators und zum Installieren von Trident benötigen. Laden Sie die neueste Version des Trident-Installationsprogramms von "der Abschnitt *Assets* auf GitHub" herunter und extrahieren Sie sie.

```
wget https://github.com/NetApp/trident/releases/download/v6.0/trident-
installer-26.02.0.tar.gz
tar -xf trident-installer-26.02.0.tar.gz
cd trident-installer
```

Schritt 2: Erstellen Sie die `TridentOrchestrator` CRD

Erstellen Sie die `TridentOrchestrator` benutzerdefinierte Ressourcendefinition (CRD). Sie werden später eine `TridentOrchestrator` benutzerdefinierte Ressource erstellen. Verwenden Sie die entsprechende CRD-YAML-Version in `deploy/crds`, um die `TridentOrchestrator` CRD zu erstellen:

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

Schritt 3: Aktualisieren Sie den Registrierungsspeicherort im Operator

In `/deploy/operator.yaml` `image: docker.io/netapp/trident-operator:26.02.0`aktualisieren Sie den Pfad, um den Speicherort Ihrer Image-Registry widerzuspiegeln. Ihre "Trident und CSI images" können sich in einer

Registry oder in verschiedenen Registrys befinden, aber alle CSI-Images müssen sich in derselben Registry befinden. Beispiel:

- `image: <your-registry>/trident-operator:26.02.0` wenn sich alle Ihre Images in einer einzigen Registry befinden.
- `image: <your-registry>/netapp/trident-operator:26.02.0`wenn sich Ihr Trident-Image in einem anderen Registry als Ihre CSI-Images befindet.

Schritt 4: Den Trident Operator einsetzen

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

- Für Cluster, auf denen Kubernetes 1.24 läuft, verwenden Sie `bundle_pre_1_25.yaml`.
- Für Cluster, auf denen Kubernetes 1.25 oder höher läuft, verwenden Sie `bundle_post_1_25.yaml`.

Bevor Sie beginnen

- Standardmäßig installiert das Trident-Installationsprogramm den Operator im `trident` Namespace. Falls der `trident` Namespace nicht existiert, erstellen Sie ihn mit:

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

- Um den Operator in einem anderen Namespace als dem `trident` Namespace bereitzustellen, aktualisieren Sie `serviceaccount.yaml`, `clusterrolebinding.yaml` und `operator.yaml` und generieren Sie Ihre Bundle-Datei mit dem `kustomization.yaml`.

  a. Erstellen Sie das `kustomization.yaml` mit dem folgenden Befehl, wobei *<bundle.yaml>* `bundle_pre_1_25.yaml` oder `bundle_post_1_25.yaml` abhängig von Ihrer Kubernetes-Version ist.

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

  b. Kompilieren Sie das Bundle mit dem folgenden Befehl, wobei *<bundle.yaml>* `bundle_pre_1_25.yaml` oder `bundle_post_1_25.yaml` entsprechend Ihrer Kubernetes-Version ist.

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

Schritte

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

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

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

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

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

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

Ihre "Trident und CSI images" können sich in einer Registry oder in verschiedenen Registrys befinden, aber alle CSI-Images müssen sich in derselben Registry befinden. Aktualisieren Sie `deploy/crds/tridentorchestrator_cr.yaml` , um die zusätzlichen Speicherortangaben entsprechend Ihrer Registry-Konfiguration hinzuzufügen.

Images in einem Registry

```
imageRegistry: "<your-registry>"
autosupportImage: "<your-registry>/trident-autosupport:26.02"
tridentImage: "<your-registry>/trident:26.02.0"
```

Images in verschiedenen Registern

```
imageRegistry: "<your-registry>"
autosupportImage: "<your-registry>/trident-autosupport:26.02"
tridentImage: "<your-registry>/trident:26.02.0"
```

Schritt 6: Erstellen Sie das `TridentOrchestrator` und installieren Sie Trident

Sie können jetzt die `TridentOrchestrator` erstellen und Trident installieren. Optional können Sie "Passen Sie Ihre Trident Installation individuell an" mithilfe der Attribute in der `TridentOrchestrator`-Spezifikation weiter anpassen. Das folgende Beispiel zeigt eine Installation, bei der sich Trident- und CSI-Images in unterschiedlichen Registries 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>/trident-autosupport:26.02
  Debug:              true
  Image Registry:     <your-registry>
  Namespace:          trident
  Trident Image:      <your-registry>/trident:26.02.0
Status:
  Current Installation Params:
    IPv6:                       false
    Autosupport Hostname:
    Autosupport Image:          <your-registry>/trident-autosupport:26.02
    Autosupport Proxy:
    Autosupport Serial Number:
    Debug:                      true
    Http Request Timeout:       90s
    Image Pull Secrets:
    Image Registry:       <your-registry>
    k8sTimeout:           30
    Kubelet Dir:          /var/lib/kubelet
    Log Format:           text
    Probe Port:           17546
    Silence Autosupport:  false
    Trident Image:        <your-registry>/trident:26.02.0
  Message:                Trident installed
  Namespace:              trident
  Status:                 Installed
  Version:                v26.02.0
Events:
    Type Reason Age From Message ---- ------ ---- ---- -------Normal
    Installing 74s trident-operator.netapp.io Installing Trident Normal
    Installed 67s trident-operator.netapp.io Trident installed
```

Überprüfen Sie die Installation

Es gibt mehrere Möglichkeiten, Ihre Installation zu überprüfen.

Status mit `TridentOrchestrator` verwenden

Der Status von `TridentOrchestrator` zeigt an, ob die Installation erfolgreich war und zeigt die installierte Version von Trident an. Während der Installation ändert sich der Status von `TridentOrchestrator` `Installing` zu `Installed`. Wenn Sie den `Failed` Status beobachten und der Operator das Problem nicht selbst beheben kann, "Überprüfen Sie die Protokolle".

Status	Beschreibung
Installation	Der Operator installiert Trident mit diesem `TridentOrchestrator` CR.
Installiert	Trident wurde erfolgreich installiert.
Deinstallation	Der Betreiber deinstalliert Trident, weil `spec.uninstall=true`.
Deinstalliert	Trident ist deinstalliert.
Fehlgeschlagen	Der Operator konnte Trident nicht installieren, patchen, aktualisieren oder deinstallieren; der Operator wird automatisch versuchen, sich von diesem Zustand zu erholen. Wenn dieser Zustand weiterhin besteht, ist eine Fehlerbehebung erforderlich.
Aktualisierung	Der Betreiber aktualisiert eine bestehende Installation.
Fehler	Das `TridentOrchestrator` wird nicht verwendet. Ein anderer existiert bereits.

Verwendung des Pod-Erstellungsstatus

Sie können bestätigen, ob die Trident Installation abgeschlossen wurde, indem Sie den Status der erstellten Pods überprüfen:

```
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
```

Verwendung `tridentctl`

Sie können `tridentctl` verwenden, um die installierte Version von Trident zu überprüfen.

```
./tridentctl -n trident version

+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 26.02.0        | 26.02.0        |
+----------------+----------------+
```

Trident Operator mit Helm bereitstellen (Standardmodus)

Sie können den Trident-Operator bereitstellen und Trident mithilfe von Helm installieren. Dieser Prozess gilt für Installationen, bei denen die von Trident benötigten Container-Images nicht in einer privaten Registry gespeichert sind. Wenn Sie eine private Image-Registry haben, verwenden Sie die "Prozess für die Offline-Bereitstellung".

Wichtige Informationen zu Trident 25.10

Sie müssen die folgenden wichtigen Informationen über Trident lesen.

Wichtige Informationen zu Trident

- Kubernetes 1.35 wird nun in Trident unterstützt. Trident sollte vor Kubernetes aktualisiert werden.
- Trident setzt die Verwendung der Multipathing-Konfiguration in SAN-Umgebungen strikt durch, mit einem empfohlenen Wert von `find_multipaths: no` in der multipath.conf-Datei.

  Die Verwendung einer Konfiguration ohne Multipathing oder die Verwendung von `find_multipaths: yes` oder `find_multipaths: smart` Wert in der Datei multipath.conf führt zu Mount-Fehlern. Trident hat die Verwendung von `find_multipaths: no` seit der Version 21.07 empfohlen.

Den Trident-Operator bereitstellen und Trident mit Helm installieren

Mit dem Trident "Helm Chart" können Sie den Trident Operator bereitstellen und Trident in einem Schritt installieren.

Überprüfen Sie "die Installationsübersicht", um sicherzustellen, dass Sie die Installationsvoraussetzungen erfüllt und die richtige Installationsoption für Ihre Umgebung ausgewählt haben.

Bevor Sie beginnen

Zusätzlich zu dem "Bereitstellungsvoraussetzungen" benötigen Sie "Helm Version 3".

Schritte

1. Fügen Sie das Trident Helm repository hinzu:

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

2. Verwenden Sie `helm install` und geben Sie einen Namen für Ihre Bereitstellung an, wie im folgenden Beispiel, wobei `100..0` die Version von Trident ist, die Sie installieren.

```
helm install <name> netapp-trident/trident-operator --version 100.2602.0
--create-namespace --namespace <trident-namespace>
```

> ⓘ Wenn Sie bereits einen Namespace für Trident erstellt haben, wird durch den `--create -namespace` Parameter kein zusätzlicher Namespace erstellt.

Sie können `helm list` verwenden, um Installationsdetails wie Name, Namespace, Chart, Status, App-Version und Revisionsnummer zu überprüfen.

Konfigurationsdaten während der Installation übergeben

Es gibt zwei Möglichkeiten, Konfigurationsdaten während der Installation zu übergeben:

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

Um beispielsweise den Standardwert von `debug` zu ändern, führen Sie den folgenden Befehl aus, wobei `100.2602.0` die Version von Trident ist, die Sie installieren:

```
helm install <name> netapp-trident/trident-operator --version 100.2602.0
--create-namespace --namespace trident --set tridentDebug=true
```

Konfigurationsoptionen

Diese Tabelle und die `values.yaml` Datei, die Teil des Helm-Charts ist, enthalten die Liste der Schlüssel und ihrer Standardwerte.

Option	Beschreibung	Standard
`nodeSelector`	Knotenbezeichnungen für die Pod-Zuweisung	
`podAnnotations`	Pod-Anmerkungen	
`deploymentAnnotations`	Bereitstellungsanmerkungen	
`tolerations`	Toleranzen für die Pod-Zuordnung	

Option	Beschreibung	Standard

`affinity`	Affinität für die Pod-Zuweisung	<pre>affinity:
		  nodeAffinity:
		
		requiredDuringSchedulingIgnoredDuringExecution:
		      nodeSelectorTerms:
		        - matchExpressions:
		            - key: kubernetes.io/arch
		              operator: In
		              values:
		              - arm64
		              - amd64
		            - key: kubernetes.io/os
		              operator: In
		              values:
		              - linux</pre>
		Entfernen Sie die Standardaffinität nicht aus der Datei values.yaml. Wenn Sie eine benutzerdefinierte Affinität angeben möchten, erweitern Sie die Standardaffinität.

`tridentControllerPluginNodeSelector`	Zusätzliche Knotenselektoren für Pods. Siehe Controller-Pods und Node-Pods verstehen für Details.	
`tridentControllerPluginTolerations`	Überschreibt die Kubernetes-Toleranzen für Pods. Siehe Controller-Pods und Node-Pods verstehen für Details.	
`tridentNodePluginNodeSelector`	Zusätzliche Knotenselektoren für Pods. Siehe Controller-Pods und Node-Pods verstehen für Details.	
`tridentNodePluginTolerations`	Überschreibt die Kubernetes-Toleranzen für Pods. Siehe Controller-Pods und Node-Pods verstehen für Details.	

Option	Beschreibung	Standard
`imageRegistry`	Identifiziert die Registry für die `trident-operator`, `trident` und andere Images. Leer lassen, um den Standardwert zu akzeptieren. WICHTIG: Wenn Sie Trident in einem privaten Repository installieren und den `imageRegistry`-Schalter zur Angabe des Repository-Speicherorts verwenden, verwenden Sie `/netapp/` nicht im Repository-Pfad.	`""`
`imagePullPolicy`	Legt die Image-Pull-Richtlinie für das `trident-operator` fest.	`IfNotPresent`
`imagePullSecrets`	Legt die Image-Pull-Secrets für die `trident-operator`, `trident` und andere Images fest.	
`kubeletDir`	Ermöglicht das Überschreiben des Host-Speicherorts des internen Zustands von kubelet.	`"/var/lib/kubelet"`
`operatorLogLevel`	Ermöglicht das Protokollierungslevel des Trident-Operators auf: `trace`, `debug`, `info`, `warn`, `error` oder `fatal` festzulegen.	`"info"`
`operatorDebug`	Ermöglicht das Festlegen des Protokollierungsgrads des Trident-Operators auf Debug.	`true`
`operatorImage`	Ermöglicht das vollständige Überschreiben des Bildes für `trident-operator`.	`""`
`operatorImageTag`	Ermöglicht das Überschreiben des Tags des `trident-operator` -Images.	`""`
`tridentIPv6`	Ermöglicht die Aktivierung von Trident für den Einsatz in IPv6-Clustern.	`false`
`tridentK8sTimeout`	Überschreibt die standardmäßige 30-Sekunden-Zeitüberschreitung für die meisten Kubernetes-API-Operationen (falls ungleich Null, in Sekunden).	`0`

Option	Beschreibung	Standard
`tridentHttpRequestTimeout`	Überschreibt die standardmäßige 90-Sekunden-Zeitüberschreitung für HTTP-Anfragen, wobei `0s` eine unbegrenzte Dauer für die Zeitüberschreitung bedeutet. Negative Werte sind nicht zulässig.	`"90s"`
`tridentSilenceAutosupport`	Ermöglicht das Deaktivieren der periodischen AutoSupport-Berichterstattung von Trident.	`false`
`tridentAutosupportImageTag`	Ermöglicht das Überschreiben des Tags des Images für den Trident AutoSupport-Container.	`<version>`
`tridentAutosupportProxy`	Ermöglicht dem Trident AutoSupport Container, über einen HTTP-Proxy nach Hause zu telefonieren.	`""`
`tridentLogFormat`	Legt das Trident Protokollierungsformat (`text` oder `json` fest.	`"text"`
`tridentDisableAuditLog`	Deaktiviert den Trident Audit-Logger.	`true`
`tridentLogLevel`	Ermöglicht das Protokollierungslevel von Trident auf `trace`, `debug`, `info`, `warn`, `error` oder `fatal` festzulegen.	`"info"`
`tridentDebug`	Ermöglicht das Festlegen des Protokollierungslevels von Trident auf `debug`. Sie können den erzwungenen Trennvorgang durch Integration mit dem node health check (NHC) Operator automatisieren. Weitere Informationen finden Sie unter "Automatisierung des Failovers von zustandsbehafteten Anwendungen mit Trident".	`false`
`tridentLogWorkflows`	Ermöglicht die Aktivierung bestimmter Trident-Workflows für die Protokollierung von Ablaufverfolgungen oder die Unterdrückung von Protokollen.	`""`
`tridentLogLayers`	Ermöglicht die Aktivierung bestimmter Trident-Ebenen für die Protokollierung von Ablaufverfolgungen oder die Unterdrückung von Protokollen.	`""`

Option	Beschreibung	Standard
`tridentImage`	Ermöglicht das vollständige Überschreiben des Images für Trident.	`""`
`tridentImageTag`	Ermöglicht das Überschreiben des Tags des Images für Trident.	`""`
`tridentProbePort`	Ermöglicht das Überschreiben des Standardports, der für Kubernetes Liveness-/Readiness-Probes verwendet wird.	`""`
`windows`	Ermöglicht die Installation von Trident auf einem Windows-Worker-Knoten.	`false`
`enableForceDetach`	Ermöglicht das Aktivieren der Funktion „Force Detach“.	`false`
`excludePodSecurityPolicy`	Schließt die Sicherheitsrichtlinie für den Operator-Pod von der Erstellung aus.	`false`
`cloudProvider`	Auf `"Azure"` setzen, wenn verwaltete Identitäten oder eine Cloud-Identität auf einem AKS-Cluster verwendet werden. Auf "AWS" setzen, wenn eine Cloud-Identität auf einem EKS-Cluster verwendet wird.	`""`
`cloudIdentity`	Legen Sie auf die Workload-Identität („azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxx“) fest, wenn Sie Cloud Identity auf einem AKS-Cluster verwenden. Legen Sie auf die AWS IAM-Rolle („'eks.amazonaws.com/role-arn: arn:aws:iam::123456:role/trident-role'“) fest, wenn Sie Cloud Identity auf einem EKS-Cluster verwenden.	`""`
`iscsiSelfHealingInterval`	Das Intervall, in dem die iSCSI Selbstreparatur aufgerufen wird.	`5m0s`
`iscsiSelfHealingWaitTime`	Die Zeitspanne, nach der die iSCSI Selbstreparatur einen Versuch unternimmt, eine veraltete Sitzung durch Durchführung einer Abmeldung und anschließenden Anmeldung zu beheben.	`7m0s`

Option	Beschreibung	Standard
`nodePrep`	Ermöglicht Trident, die Knoten des Kubernetes-Clusters für die Verwaltung von Volumes mithilfe des angegebenen Datenspeicherprotokolls vorzubereiten. **Derzeit ist `iscsi` der einzige unterstützte Wert.** HINWEIS: Ab OpenShift 4.19 ist die Mindestversion von Trident für diese Funktion 25.06.1.	
`enableConcurrency`	Ermöglicht gleichzeitige Trident-Controller-Operationen für einen verbesserten Durchsatz. **Tech Preview**: Diese Funktion ist experimentell und unterstützt derzeit eingeschränkte parallele Arbeitsabläufe mit den ONTAP-NAS (nur NFS) und ONTAP-SAN (NVMe für unified ONTAP 9) Treibern, zusätzlich zur bestehenden Tech Preview für den ONTAP-SAN Treiber (iSCSI- und FCP-Protokolle in unified ONTAP 9).	false
`k8sAPIQPS`	Die vom Controller bei der Kommunikation mit dem Kubernetes-API-Server verwendete Abfrageanzahl pro Sekunde (QPS). Der Burst-Wert wird automatisch anhand des QPS-Werts festgelegt.	`100`; optional

Option	Beschreibung	Standard

`resources`	Legt Kubernetes-Ressourcenlimits und -anforderungen für die Trident-Controller-, Node- und Operator-Pods fest. Sie können CPU und Arbeitsspeicher für jeden Container und Sidecar konfigurieren, um die Ressourcenzuweisung in Kubernetes zu verwalten.
	
	Weitere Informationen zur Konfiguration von Ressourcenanforderungen und -limits finden Sie unter "Ressourcenverwaltung für Pods und Container".
	
	• Ändern Sie NICHT die Namen von Containern oder Feldern.
	• Ändern Sie die Einrückung NICHT – die YAML-Einrückung ist für das korrekte Parsen entscheidend.
		<pre>resources:
		  controller:
		    trident-main:
		      requests:
		        cpu: 10m
		        memory: 80Mi
		      limits:
		          cpu:
		          memory:
		    csi-provisioner:
		      requests:
		        cpu: 2m
		        memory: 20Mi
		      limits:
		        cpu:
		        memory:
		    csi-attacher:
		      requests:
		        cpu: 2m
		        memory: 20Mi
		      limits:
		        cpu:
		        memory:
		    csi-resizer:
		      requests:
		        cpu: 3m
		        memory: 20Mi
		      limits:
		        cpu:
		        memory:
		    csi-snapshotter:
		      requests:
		        cpu: 2m
		        memory: 20Mi
		      limits:
		        cpu:
		        memory:
		    trident-autosupport:
		      requests:
		        cpu: 1m
		        memory: 30Mi
		      limits:
		        cpu:
		        memory:
		  node:
		    linux:
		      trident-main:</pre>

Option	Beschreibung	Standard
`httpsMetrics`	Aktivieren Sie HTTPS für den Prometheus metrics endpoint.	false
`hostNetwork`	Aktiviert die Host-Netzwerkverbindung für den Trident Controller. Dies ist nützlich, wenn Sie den Frontend- und Backend-Datenverkehr in einem Multi-Home-Netzwerk trennen möchten.	false

wenn sie nicht angegeben sind.

- Containernamen werden so aufgelistet, wie sie in den Pod-Spezifikationen erscheinen.
- Sidecars sind unter jedem Hauptcontainer aufgeführt.
- Überprüfen Sie das Feld „TORC“ `status.CurrentInstallationParams`, um die aktuell angewendeten Werte anzuzeigen.

```
            memory: 10Mi
        limits:
          cpu:
          memory:
      windows:
        trident-main:
        requests:
          cpu: 6m
          memory: 40Mi
        limits:
          cpu:
          memory:
        node-driver-registrar:
        requests:
          cpu: 6m
          memory: 40Mi
        limits:
          cpu:
          memory:
        liveness-probe:
          requests:
            cpu: 2m
            memory: 40Mi
          limits:
            cpu:
            memory:
    operator:
      requests:
        cpu: 10m
        memory: 40Mi
      limits:
        cpu:
        memory:
```

Controller-Pods und Node-Pods verstehen

Trident läuft als einzelner Controller-Pod sowie als Node-Pod auf jedem Worker-Knoten im Cluster. Der Node-Pod muss auf jedem Host ausgeführt werden, auf dem Sie möglicherweise ein Trident Volume einbinden möchten.

Kubernetes "Knotenselektoren" und "Tolerations und Taints" werden verwendet, um einen Pod auf einen bestimmten oder bevorzugten Knoten zu beschränken. Mit dem `ControllerPlugin` und `NodePlugin` können Sie Einschränkungen und Überschreibungen festlegen.

- Das Controller-Plugin übernimmt die Bereitstellung und Verwaltung von Volumes, wie Snapshots und Größenänderungen.
- Das Node-Plugin übernimmt das Anbinden des Speichers an den Knoten.

Trident operator mit Helm bereitstellen (Offline mode)

Sie können den Trident-Operator bereitstellen und Trident mithilfe von Helm installieren. Dieser Vorgang gilt für Installationen, bei denen die von Trident benötigten Container-Images in einer privaten Registry gespeichert sind. Wenn Sie keine private Image-Registry besitzen, verwenden Sie die "Prozess für die Standardbereitstellung".

Wichtige Informationen zu Trident 26.02

Sie müssen die folgenden wichtigen Informationen über Trident lesen.

Wichtige Informationen zu Trident

- Kubernetes 1.35 wird nun in Trident unterstützt. Trident sollte vor Kubernetes aktualisiert werden.
- Trident setzt die Verwendung der Multipathing-Konfiguration in SAN-Umgebungen strikt durch, mit einem empfohlenen Wert von `find_multipaths: no` in der multipath.conf-Datei.

  Die Verwendung einer Konfiguration ohne Multipathing oder die Verwendung von `find_multipaths: yes` oder `find_multipaths: smart` Wert in der Datei multipath.conf führt zu Mount-Fehlern. Trident hat die Verwendung von `find_multipaths: no` seit der Version 21.07 empfohlen.

Den Trident-Operator bereitstellen und Trident mit Helm installieren

Mit dem Trident "Helm Chart" können Sie den Trident Operator bereitstellen und Trident in einem Schritt installieren.

Überprüfen Sie "die Installationsübersicht", um sicherzustellen, dass Sie die Installationsvoraussetzungen erfüllt und die richtige Installationsoption für Ihre Umgebung ausgewählt haben.

Bevor Sie beginnen

Zusätzlich zu dem "Bereitstellungsvoraussetzungen" benötigen Sie "Helm Version 3".

> Wenn Sie Trident in einem privaten Repository installieren und den `imageRegistry` Schalter verwenden, um den Speicherort des Repositorys anzugeben, verwenden Sie `/netapp/` nicht im Repository-Pfad.

Schritte

1. Fügen Sie das Trident Helm repository hinzu:

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

2. Verwenden Sie `helm install` und geben Sie einen Namen für Ihre Bereitstellung sowie den Speicherort der Image-Registry an. Ihre "Trident und CSI images" können sich in einer Registry oder in verschiedenen Registrys befinden, aber alle CSI-Images müssen sich in derselben Registry befinden. In den Beispielen ist `100.2602.0` die Version von Trident, die Sie installieren.

Images in einem Registry

```
helm install <name> netapp-trident/trident-operator --version
100.2602.0 --set imageRegistry=<your-registry> --create-namespace
--namespace <trident-namespace> --set nodePrep={iscsi}
```

Images in verschiedenen Registern

```
helm install <name> netapp-trident/trident-operator --version
100.2602.0 --set imageRegistry=<your-registry> --set operatorImage
=<your-registry>/trident-operator:26.02.0 --set
tridentAutosupportImage=<your-registry>/trident-autosupport:26.02
--set tridentImage=<your-registry>/trident:26.02.0 --create
-namespace --namespace <trident-namespace> --set nodePrep={iscsi}
```

Wenn Sie bereits einen Namespace für Trident erstellt haben, wird durch den `--create -namespace` Parameter kein zusätzlicher Namespace erstellt.

Sie können `helm list` verwenden, um Installationsdetails wie Name, Namespace, Chart, Status, App-Version und Revisionsnummer zu überprüfen.

Konfigurationsdaten während der Installation übergeben

Es gibt zwei Möglichkeiten, Konfigurationsdaten während der Installation zu übergeben:

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

Um beispielsweise den Standardwert von `debug` zu ändern, führen Sie den folgenden Befehl aus, wobei `100.2602.0` die Version von Trident ist, die Sie installieren:

```
helm install <name> netapp-trident/trident-operator --version 100.2602.0
--create-namespace --namespace trident --set tridentDebug=true
```

Um den nodePrep Wert hinzuzufügen, führen Sie den folgenden Befehl aus:

```
helm install <name> netapp-trident/trident-operator --version 100.2602.0
--create-namespace --namespace trident --set nodePrep={iscsi}
```

Konfigurationsoptionen

Diese Tabelle und die `values.yaml` Datei, die Teil des Helm-Charts ist, enthalten die Liste der Schlüssel und ihrer Standardwerte.

> Entfernen Sie die Standardaffinität nicht aus der Datei values.yaml. Wenn Sie eine benutzerdefinierte Affinität angeben möchten, erweitern Sie die Standardaffinität.

Option	Beschreibung	Standard
`nodeSelector`	Knotenbezeichnungen für die Pod-Zuweisung	
`podAnnotations`	Pod-Anmerkungen	
`deploymentAnnotations`	Bereitstellungsanmerkungen	
`tolerations`	Toleranzen für die Pod-Zuordnung	

Option	Beschreibung	Standard

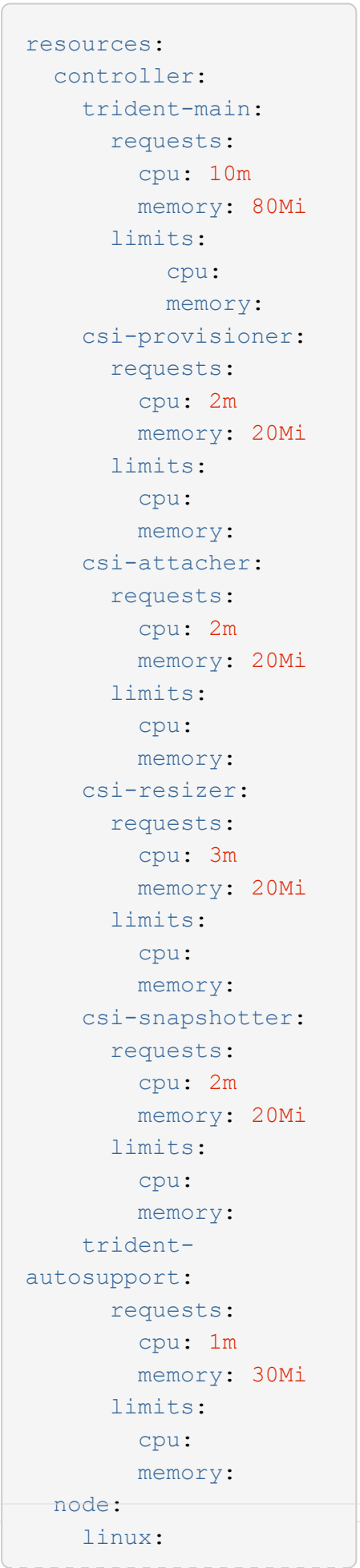
`affinity`	Affinität für die Pod-Zuweisung	`affinity:`
		`  nodeAffinity:`
		
		`requiredDuringSchedul`
		`ingIgnoredDuringExecu`
		`tion:`
		
		`nodeSelectorTerms:`
		`  -`
		`matchExpressions:`
		`  - key:`
		`kubernetes.io/arch`
		
		`operator: In`
		`  values:`
		`  - arm64`
		`  - amd64`
		`  - key:`
		`kubernetes.io/os`
		
		`operator: In`
		`  values:`
		`  - linux`
		
		Entfernen Sie die Standardaffinität nicht aus der Datei values.yaml. Wenn Sie eine benutzerdefinierte Affinität angeben möchten, erweitern Sie die Standardaffinität.

`tridentControllerPluginNod` `eSelector`	Zusätzliche Knotenselektoren für Pods. Siehe "Controller-Pods und Node-Pods verstehen" für Details.	
`tridentControllerPluginTol` `erations`	Überschreibt die Kubernetes-Toleranzen für Pods. Siehe "Controller-Pods und Node-Pods verstehen" für Details.	

Option	Beschreibung	Standard
`tridentNodePluginNodeSelector`	Zusätzliche Knotenselektoren für Pods. Siehe "Controller-Pods und Node-Pods verstehen" für Details.	
`tridentNodePluginTolerations`	Überschreibt die Kubernetes-Toleranzen für Pods. Siehe "Controller-Pods und Node-Pods verstehen" für Details.	
`imageRegistry`	Identifiziert die Registry für die `trident-operator`, `trident` und andere Images. Leer lassen, um den Standardwert zu akzeptieren. WICHTIG: Wenn Sie Trident in einem privaten Repository installieren und den `imageRegistry`-Schalter zur Angabe des Repository-Speicherorts verwenden, verwenden Sie `/netapp/` nicht im Repository-Pfad.	""
`imagePullPolicy`	Legt die Image-Pull-Richtlinie für das `trident-operator` fest.	`IfNotPresent`
`imagePullSecrets`	Legt die Image-Pull-Secrets für die `trident-operator`, `trident` und andere Images fest.	
`kubeletDir`	Ermöglicht das Überschreiben des Host-Speicherorts des internen Zustands von kubelet.	`"/var/lib/kubelet"`
`operatorLogLevel`	Ermöglicht das Protokollierungslevel des Trident-Operators auf: `trace`, `debug`, `info`, `warn`, `error` oder `fatal` festzulegen.	`"info"`
`operatorDebug`	Ermöglicht das Festlegen des Protokollierungsgrads des Trident-Operators auf Debug.	`true`
`operatorImage`	Ermöglicht das vollständige Überschreiben des Bildes für `trident-operator`.	""
`operatorImageTag`	Ermöglicht das Überschreiben des Tags des `trident-operator` -Images.	""
`tridentIPv6`	Ermöglicht die Aktivierung von Trident für den Einsatz in IPv6-Clustern.	`false`

Option	Beschreibung	Standard
`tridentK8sTimeout`	Überschreibt die standardmäßige 180-Sekunden-Zeitüberschreitung für die meisten Kubernetes-API-Operationen (falls ungleich Null, in Sekunden). Der `tridentK8sTimeout` Parameter ist nur für Trident-Installationen anwendbar.	`180`
`tridentHttpRequestTimeout`	Überschreibt die standardmäßige 90-Sekunden-Zeitüberschreitung für HTTP-Anfragen, wobei `0s` eine unbegrenzte Dauer für die Zeitüberschreitung bedeutet. Negative Werte sind nicht zulässig.	`"90s"`
`tridentSilenceAutosupport`	Ermöglicht das Deaktivieren der periodischen AutoSupport-Berichterstattung von Trident.	`false`
`tridentAutosupportImageTag`	Ermöglicht das Überschreiben des Tags des Images für den Trident AutoSupport-Container.	`<version>`
`tridentAutosupportProxy`	Ermöglicht dem Trident AutoSupport Container, über einen HTTP-Proxy nach Hause zu telefonieren.	`""`
`tridentLogFormat`	Legt das Trident Protokollierungsformat (`text` oder `json` fest.	`"text"`
`tridentDisableAuditLog`	Deaktiviert den Trident Audit-Logger.	`true`
`tridentLogLevel`	Ermöglicht das Protokollierungslevel von Trident auf `trace`, `debug`, `info`, `warn`, `error` oder `fatal` festzulegen.	`"info"`
`tridentDebug`	Ermöglicht das Festlegen des Protokollierungslevels von Trident auf `debug`.	`false`
`tridentLogWorkflows`	Ermöglicht die Aktivierung bestimmter Trident-Workflows für die Protokollierung von Ablaufverfolgungen oder die Unterdrückung von Protokollen.	`""`

Option	Beschreibung	Standard
`tridentLogLayers`	Ermöglicht die Aktivierung bestimmter Trident-Ebenen für die Protokollierung von Ablaufverfolgungen oder die Unterdrückung von Protokollen.	""
`tridentImage`	Ermöglicht das vollständige Überschreiben des Images für Trident.	""
`tridentImageTag`	Ermöglicht das Überschreiben des Tags des Images für Trident.	""
`tridentProbePort`	Ermöglicht das Überschreiben des Standardports, der für Kubernetes Liveness-/Readiness-Probes verwendet wird.	""
`windows`	Ermöglicht die Installation von Trident auf einem Windows-Worker-Knoten.	`false`
`enableForceDetach`	Ermöglicht das Aktivieren der Funktion „Force Detach“. Sie können den erzwungenen Trennvorgang durch Integration mit dem node health check (NHC) Operator automatisieren. Weitere Informationen finden Sie unter "Automatisierung des Failovers von zustandsbehafteten Anwendungen mit Trident".	`false`
`excludePodSecurityPolicy`	Schließt die Sicherheitsrichtlinie für den Operator-Pod von der Erstellung aus.	`false`
`nodePrep`	Ermöglicht Trident, die Knoten des Kubernetes-Clusters für die Verwaltung von Volumes mithilfe des angegebenen Datenspeicherprotokolls vorzubereiten. **Derzeit ist `iscsi` der einzige unterstützte Wert.** Ab OpenShift 4.19 ist die minimale unterstützte Trident-Version für diese Funktion 25.06.1.	

Option	Beschreibung	Standard
`resources`	Legt Kubernetes-Ressourcenlimits und -anforderungen für die Trident-Controller-, Node- und Operator-Pods fest. Sie können CPU und Arbeitsspeicher für jeden Container und Sidecar konfigurieren, um die Ressourcenzuweisung in Kubernetes zu verwalten. Weitere Informationen zur Konfiguration von Ressourcenanforderungen und -limits finden Sie unter "Ressourcenverwaltung für Pods und Container". • Ändern Sie NICHT die Namen von Containern oder Feldern. • Ändern Sie die Einrückung NICHT – die YAML-Einrückung ist für das korrekte Parsen entscheidend.	

Trident-Operator-Installation anpassen

Der Trident-Operator ermöglicht es Ihnen, die Trident-Installation mithilfe der Attribute in der `TridentOrchestrator` spec anzupassen. Wenn Sie die Installation über das hinaus anpassen möchten, was `TridentOrchestrator` Argumente erlauben, sollten Sie `tridentctl` verwenden, um benutzerdefinierte YAML-Manifeste zu generieren und diese nach Bedarf zu modifizieren.

Controller-Pods und Node-Pods verstehen

Trident läuft als einzelner Controller-Pod und als Node-Pod auf jedem Worker-Knoten im Cluster. Der Node-Pod muss auf jedem Host ausgeführt werden, auf dem Sie möglicherweise ein Trident Volume einbinden möchten.

Kubernetes "Knotenselektoren" und "Tolerations und Taints" werden verwendet, um einen Pod auf einen bestimmten oder bevorzugten Knoten zu beschränken. Mit dem `ControllerPlugin` und `NodePlugin` können Sie Einschränkungen und Überschreibungen festlegen.

- Das Controller-Plugin übernimmt die Bereitstellung und Verwaltung von Volumes, wie Snapshots und Größenänderungen.
- Das Node-Plugin übernimmt das Anbinden des Speichers an den Knoten.

Konfigurationsoptionen

> `spec.namespace` wird in `TridentOrchestrator` angegeben, um den Namespace zu kennzeichnen, in dem Trident installiert ist. Dieser Parameter **kann nach der Installation von Trident nicht mehr aktualisiert werden**. Ein entsprechender Versuch führt dazu, dass sich der `TridentOrchestrator` Status in `Failed` ändert. Trident ist nicht für die Migration zwischen Namespaces vorgesehen.

Diese Tabelle enthält `TridentOrchestrator`Attribute im Detail.

Parameter	Beschreibung	Standard
`namespace`	Namespace, in dem Trident installiert werden soll	`"default"`
`debug`	Debugging für Trident aktivieren	`false`
`enableForceDetach`	`ontap-san`, `ontap-san-economy`, `ontap-nas`, und `ontap-nas-economy` nur. Funktioniert mit Kubernetes Non-Graceful Node Shutdown (NGNS), um Clusteradministratoren die Möglichkeit zu geben, Workloads mit eingebundenen Volumes sicher auf neue Knoten zu migrieren, falls ein Knoten fehlerhaft wird. Weitere Informationen finden Sie unter "Automatisierung des Failovers von zustandsbehafteten Anwendungen mit Trident".	`false`
`windows`	Das Setzen auf `true` ermöglicht die Installation auf Windows-Worker-Knoten.	`false`

- Es gelten standardmäßig keine Beschränkungen für Requests haben Standardwerte.
- Containernamen werden so aufgelistet, wie sie in den Pod-Spezifikationen erscheinen.
- Sidecars sind unter jedem Hauptcontainer aufgeführt.
- Überprüfen Sie das Feld `TORC` `status.CurrentInstallationParams`, um die aktuell angewendeten Werte anzuzeigen.

```
tridentNode:
  requests:
    cpu: 10m
    memory: 60Mi
  limits:
    cpu:
    memory:
  nodeDriverRegistrar:
  requests:
    cpu: 1m
    memory: 10Mi
  limits:
    cpu:
    memory:
windows:
  trident-main:
    requests:
      cpu: 6m
      memory: 40Mi
    limits:
      cpu:
      memory:
  node-driver-registrar:
      memory:
operator:
  requests:
    cpu: 10m
    memory: 40Mi
  limits:
```

Parameter	Beschreibung	Standard
`cloudProvider`	Auf `"Azure"` setzen, wenn verwaltete Identitäten oder eine Cloud-Identität auf einem AKS-Cluster verwendet werden. Setzen Sie auf `"AWS"`, wenn eine Cloud-Identität auf einem EKS-Cluster verwendet wird. Setzen Sie auf `"GCP"`, wenn eine Cloud-Identität auf einem GKE-Cluster verwendet wird.	`""`
`cloudIdentity`	Legen Sie auf die Workload-Identität („azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxx“) fest, wenn Sie Cloud Identity auf einem AKS-Cluster verwenden. Legen Sie auf die AWS IAM-Rolle („'eks.amazonaws.com/role-arn: arn:aws:iam::123456:role/trident-role'“) fest, wenn Sie Cloud Identity auf einem EKS-Cluster verwenden. Auf Cloud-Identität („'iam.gke.io/gcp-service-account: xxxx@mygcpproject.iam.gserviceaccount.com'“) setzen, wenn Cloud Identity auf einem GKE-Cluster verwendet wird.	`""`
`IPv6`	Trident über IPv6 installieren	false
`k8sTimeout`	Zeitüberschreitung für Kubernetes-Operationen. Der `k8sTimeout` Parameter ist nur für Trident-Installationen anwendbar.	`180sec`
`silenceAutosupport`	Senden Sie Autosupport-Pakete nicht automatisch an NetApp	`false`
`autosupportImage`	Das Container-Image für Autosupport Telemetry	`"netapp/trident-autosupport10"`
`autosupportProxy`	Die Adresse/der Port eines Proxys zum Senden von Autosupport-Telemetrie	`"http://proxy.example.com:8888"`
`uninstall`	Eine Kennzeichnung, die zur Deinstallation von Trident verwendet wird	`false`
`logFormat`	Trident Protokollierungsformat, das verwendet werden soll [text,json]	`"text"`
`tridentImage`	Trident-Image zur Installation	`"netapp/trident:26.02"`
`imageRegistry`	Pfad zur internen Registry im Format `<registry FQDN>[:port][/subpath]`	`"registry.k8s.io"`
`kubeletDir`	Pfad zum kubelet-Verzeichnis auf dem Host	`"/var/lib/kubelet"`
`wipeout`	Eine Liste der Ressourcen, die gelöscht werden müssen, um eine vollständige Entfernung von Trident durchzuführen	

Parameter	Beschreibung	Standard
`imagePullSecrets`	Geheimnisse zum Abrufen von Images aus einer internen Registry	
`imagePullPolicy`	Legt die Richtlinie zum Abrufen von Images für den Trident Operator fest. Gültige Werte sind: `Always` um das Image immer abzurufen. `IfNotPresent` um das Image nur abzurufen, wenn es noch nicht auf dem Knoten vorhanden ist. `Never` um das Image niemals abzurufen.	`IfNotPresent`
`controllerPluginNodeSelector`	Zusätzliche Knotenselektoren für Pods. Folgt dem gleichen Format wie `pod.spec.nodeSelector`.	Kein Standardwert; optional
`controllerPluginTolerations`	Überschreibt die Kubernetes-Toleranzen für Pods. Folgt dem gleichen Format wie `pod.spec.Tolerations`.	Kein Standardwert; optional
`nodePluginNodeSelector`	Zusätzliche Knotenselektoren für Pods. Folgt dem gleichen Format wie `pod.spec.nodeSelector`.	Kein Standardwert; optional
`nodePluginTolerations`	Überschreibt die Kubernetes-Toleranzen für Pods. Folgt dem gleichen Format wie `pod.spec.Tolerations`.	Kein Standardwert; optional
`nodePrep`	Ermöglicht Trident, die Knoten des Kubernetes-Clusters für die Verwaltung von Volumes mithilfe des angegebenen Datenspeicherprotokolls vorzubereiten. **Derzeit ist `iscsi` der einzige unterstützte Wert.** Ab OpenShift 4.19 ist die minimale unterstützte Trident-Version für diese Funktion 25.06.1.	
`k8sAPIQPS`	Die vom Controller bei der Kommunikation mit dem Kubernetes-API-Server verwendete Abfrageanzahl pro Sekunde (QPS). Der Burst-Wert wird automatisch anhand des QPS-Werts festgelegt.	`100`; optional
`enableConcurrency`	Ermöglicht gleichzeitige Trident-Controller-Operationen für einen verbesserten Durchsatz. **Tech Preview**: Diese Funktion ist experimentell und unterstützt derzeit eingeschränkte parallele Arbeitsabläufe mit den ONTAP-NAS (nur NFS) und ONTAP-SAN (NVMe für unified ONTAP 9) Treibern, zusätzlich zur bestehenden Tech Preview für den ONTAP-SAN Treiber (iSCSI- und FCP-Protokolle in unified ONTAP 9).	false

Parameter	Beschreibung	Standard
`resources`	Legt Ressourcenlimits und -anforderungen für den Trident-Controller und die Node-Pods in Kubernetes fest. Sie können CPU und Arbeitsspeicher für jeden Container und Sidecar konfigurieren, um die Ressourcenzuweisung in Kubernetes zu verwalten. Weitere Informationen zur Konfiguration von Ressourcenanforderungen und -limits finden Sie unter "Ressourcenverwaltung für Pods und Container". • Ändern Sie NICHT die Namen von Containern oder Feldern. • Ändern Sie die Einrückung NICHT – die YAML-Einrückung ist für das korrekte Parsen entscheidend. • Standardmäßig werden keine Limits angewendet – nur Anfragen haben Standardwerte und werden automatisch angewendet, wenn sie nicht angegeben sind. • Containernamen werden so aufgelistet, wie sie in den Pod-Spezifikationen erscheinen. • Sidecars sind unter jedem Hauptcontainer aufgeführt. • Überprüfen Sie das Feld „TORC“ `status.CurrentInstallationParams`, um die aktuell angewendeten Werte anzuzeigen.	`resources: controller: trident-main: requests: cpu: 10m memory: 80Mi limits: cpu: memory: csi-provisioner: requests: cpu: 2m memory: 20Mi limits: cpu: memory: csi-attacher: requests: cpu: 2m memory: 20Mi limits: cpu: memory: csi-resizer: requests: cpu: 3m memory: 20Mi limits: cpu: memory: csi-snapshotter: requests: cpu: 2m memory: 20Mi limits:`

Parameter	Beschreibung	Standard
`httpsMetrics`	Aktivieren Sie HTTPS für den Prometheus metrics endpoint.	false
`hostNetwork`	Aktiviert die Host-Netzwerkverbindung für den Trident Controller. Dies ist nützlich, wenn Sie den Frontend- und Backend-Datenverkehr in einem Multi-Home-Netzwerk trennen möchten.	false

Weitere Informationen zur Formatierung von Pod-Parametern finden Sie unter "Pods.Knoten zuweisen".

Beispielkonfigurationen

Sie können die Attribute in Konfigurationsoptionen verwenden, wenn Sie `TridentOrchestrator` definieren, um Ihre Installation anzupassen.

Grundlegende benutzerdefinierte Konfiguration

Dieses Beispiel, das nach Ausführung des `cat deploy/crds/tridentorchestrator_cr_imagepullsecrets.yaml` Befehls erstellt wurde, stellt eine einfache benutzerdefinierte Installation dar:

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

Knotenselektoren

Dieses Beispiel installiert Trident mit Knotenselektoren.

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

Windows-Worker-Knoten

Dieses Beispiel, das nach Ausführung des `cat deploy/crds/tridentorchestrator_cr.yaml` Befehls erstellt wurde, installiert Trident auf einem Windows-Worker-Knoten.

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  windows: true
```

Verwaltete Identitäten auf einem AKS-Cluster

Dieses Beispiel installiert Trident, um verwaltete Identitäten auf einem AKS-Cluster zu aktivieren.

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  cloudProvider: "Azure"
```

Cloud-Identität auf einem AKS-Cluster

Dieses Beispiel installiert Trident zur Verwendung mit einer Cloud-Identität auf einem AKS-Cluster.

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  cloudProvider: "Azure"
  cloudIdentity: 'azure.workload.identity/client-id: xxxxxxxx-xxxx-
xxxx-xxxx-xxxxxxxxxxx'
```

Cloud-Identität auf einem EKS-Cluster

Dieses Beispiel installiert Trident zur Verwendung mit einer Cloud-Identität auf einem AKS-Cluster.

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  cloudProvider: "AWS"
  cloudIdentity: "'eks.amazonaws.com/role-arn:
arn:aws:iam::123456:role/trident-role'"
```

Cloud-Identität für GKE

Dieses Beispiel installiert Trident zur Verwendung mit einer Cloud-Identität auf einem GKE-Cluster.

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-gcp-gcnv
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: '012345678901'
  network: gcnv-network
  location: us-west2
  serviceLevel: Premium
  storagePool: pool-premium1
```

Konfiguration der Kubernetes-Ressourcenanforderungen und -limits für Trident controller und Trident Linux node pods

Dieses Beispiel konfiguriert Kubernetes-Ressourcenanforderungen und -limits für Trident controller und Trident Linux node pods.

Hinweis: Die in diesem Beispiel angegebenen Anforderungs- und Grenzwerte dienen lediglich Demonstrationszwecken. Passen Sie diese Werte an Ihre Umgebung und Arbeitslastanforderungen an.

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  imagePullSecrets:
  - thisisasecret
  resources:
    controller:
      trident-main:
        requests:
          cpu: 10m
          memory: 80Mi
        limits:
          cpu: 200m
          memory: 256Mi
      # sidecars
      csi-provisioner:
        requests:
          cpu: 2m
          memory: 20Mi
        limits:
          cpu: 100m
          memory: 64Mi
      csi-attacher:
        requests:
          cpu: 2m
          memory: 20Mi
        limits:
          cpu: 100m
          memory: 64Mi
      csi-resizer:
        requests:
          cpu: 3m
```

```
          memory: 20Mi
        limits:
          cpu: 100m
          memory: 64Mi
      csi-snapshotter:
        requests:
          cpu: 2m
          memory: 20Mi
        limits:
          cpu: 100m
          memory: 64Mi
      trident-autosupport:
        requests:
          cpu: 1m
          memory: 30Mi
        limits:
          cpu: 50m
          memory: 128Mi
    node:
      linux:
        trident-main:
          requests:
            cpu: 10m
            memory: 60Mi
          limits:
            cpu: 200m
            memory: 256Mi
        # sidecars
        node-driver-registrar:
          requests:
            cpu: 1m
            memory: 10Mi
          limits:
            cpu: 50m
            memory: 32Mi
```

Konfiguration von Kubernetes-Ressourcenanforderungen und -limits für Trident Controller sowie für Trident Windows- und Linux-Node-Pods

Dieses Beispiel konfiguriert Kubernetes-Ressourcenanforderungen und -Limits für Trident Controller sowie Trident Windows- und Linux Node Pods.

Hinweis: Die in diesem Beispiel angegebenen Anforderungs- und Grenzwerte dienen lediglich Demonstrationszwecken. Passen Sie diese Werte an Ihre Umgebung und Arbeitslastanforderungen an.

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  imagePullSecrets:
  - thisisasecret
  windows: true
  resources:
    controller:
      trident-main:
        requests:
          cpu: 10m
          memory: 80Mi
        limits:
          cpu: 200m
          memory: 256Mi
        # sidecars
      csi-provisioner:
        requests:
          cpu: 2m
          memory: 20Mi
        limits:
          cpu: 100m
          memory: 64Mi
      csi-attacher:
        requests:
          cpu: 2m
          memory: 20Mi
        limits:
          cpu: 100m
          memory: 64Mi
      csi-resizer:
        requests:
```

```
          cpu: 3m
          memory: 20Mi
        limits:
          cpu: 100m
          memory: 64Mi
      csi-snapshotter:
        requests:
          cpu: 2m
          memory: 20Mi
        limits:
          cpu: 100m
          memory: 64Mi
      trident-autosupport:
        requests:
          cpu: 1m
          memory: 30Mi
        limits:
          cpu: 50m
          memory: 128Mi
    node:
      linux:
        trident-main:
          requests:
            cpu: 10m
            memory: 60Mi
          limits:
            cpu: 200m
            memory: 256Mi
        # sidecars
        node-driver-registrar:
          requests:
            cpu: 1m
            memory: 10Mi
          limits:
            cpu: 50m
            memory: 32Mi
      windows:
        trident-main:
          requests:
            cpu: 6m
            memory: 40Mi
          limits:
            cpu: 200m
            memory: 128Mi
        # sidecars
        node-driver-registrar:
```

```
      requests:
        cpu: 6m
        memory: 40Mi
      limits:
        cpu: 100m
        memory: 128Mi
    liveness-probe:
      requests:
        cpu: 2m
        memory: 40Mi
      limits:
        cpu: 50m
        memory: 64Mi
```

Installation mit tridentctl

Installation mit tridentctl

Sie können Trident mit `tridentctl` installieren. Dieser Prozess gilt für Installationen, bei denen die von Trident benötigten Container-Images entweder in einer privaten Registry gespeichert sind oder nicht. Um Ihre `tridentctl` Bereitstellung anzupassen, siehe "tridentctl-Bereitstellung anpassen".

Wichtige Informationen zu Trident10

Sie müssen die folgenden wichtigen Informationen über Trident lesen.

Wichtige Informationen zu Trident

- Kubernetes 1.27 wird jetzt in Trident unterstützt. Trident sollte vor Kubernetes aktualisiert werden.
- Trident setzt die Verwendung der Multipathing-Konfiguration in SAN-Umgebungen strikt durch, mit einem empfohlenen Wert von `find_multipaths: no` in der multipath.conf-Datei.

  Die Verwendung einer Konfiguration ohne Multipathing oder die Verwendung von `find_multipaths: yes` oder `find_multipaths: smart` Wert in der Datei multipath.conf führt zu Mount-Fehlern. Trident hat die Verwendung von `find_multipaths: no` seit der Version 21.07 empfohlen.

Installieren Sie Trident mit `tridentctl`

Überprüfen Sie "die Installationsübersicht", um sicherzustellen, dass Sie die Installationsvoraussetzungen erfüllt und die richtige Installationsoption für Ihre Umgebung ausgewählt haben.

Bevor Sie beginnen

Bevor Sie mit der Installation beginnen, melden Sie sich am Linux-Host an und vergewissern Sie sich, dass dieser ein funktionierendes "unterstützter Kubernetes-Cluster" verwaltet und dass Sie über die erforderlichen

Berechtigungen verfügen.

> Mit OpenShift verwenden Sie `oc` statt `kubectl` in allen folgenden Beispielen und melden Sie sich zuerst als **system:admin** an, indem Sie `oc login -u system:admin` oder `oc login -u kube-admin` ausführen.

1. Überprüfen Sie Ihre Kubernetes-Version:

```
kubectl version
```

2. Überprüfen Sie die Cluster-Administratorrechte:

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

3. Überprüfen Sie, ob Sie einen Pod starten können, der ein Image von Docker Hub verwendet, und Ihr Speichersystem über das Pod-Netzwerk erreichen können.

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

Schritt 1: Laden Sie das Trident-Installationspaket herunter

Das Trident-Installationspaket erstellt einen Trident Pod, konfiguriert die CRD-Objekte, die zur Zustandsverwaltung verwendet werden, und initialisiert die CSI-Sidecars, um Aktionen wie das Bereitstellen und Anbinden von Volumes an die Cluster-Hosts auszuführen. Laden Sie die neueste Version des Trident-Installationsprogramms von "der Abschnitt *Assets* auf GitHub" herunter und extrahieren Sie sie. Aktualisieren Sie *<trident-installer-XX.XX.X.tar.gz>* im Beispiel mit Ihrer ausgewählten Trident Version.

```
wget https://github.com/NetApp/trident/releases/download/v26.02.0/trident-
installer-26.02.0.tar.gz
tar -xf trident-installer-26.02.0.tar.gz
cd trident-installer
```

Schritt 2: Trident installieren

Installieren Sie Trident im gewünschten Namespace, indem Sie den `tridentctl install` Befehl ausführen. Sie können zusätzliche Argumente hinzufügen, um den Speicherort der Image-Registry anzugeben.

Standardmodus

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

Images in einem Registry

```
./tridentctl install -n trident --image-registry <your-registry>
--autosupport-image <your-registry>/trident-autosupport:26.02 --trident
-image <your-registry>/trident:26.02.0
```

Images in verschiedenen Registern

```
./tridentctl install -n trident --image-registry <your-registry>
--autosupport-image <your-registry>/trident-autosupport:26.02 --trident
-image <your-registry>/trident:26.02.0
```

Ihr Installationsstatus sollte in etwa 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=26.10.0
INFO Trident installation succeeded.
....
```

Überprüfen Sie die Installation

Sie können Ihre Installation anhand des pod creation status oder `tridentctl` überprüfen.

Verwendung des Pod-Erstellungsstatus

Sie können bestätigen, ob die Trident Installation abgeschlossen wurde, indem Sie den Status der erstellten Pods überprüfen:

```
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
```

Falls das Installationsprogramm nicht erfolgreich abgeschlossen wird oder `trident-controller-<generated id>` (`trident-csi-<generated id>` in Versionen vor 23.01) keinen **Running**-Status hat, wurde die Plattform nicht installiert. Verwenden Sie `-d` , um "Debug-Modus aktivieren" das Problem zu beheben und zu analysieren.

Verwendung `tridentctl`

Sie können `tridentctl` verwenden, um die installierte Version von Trident zu überprüfen.

```
./tridentctl -n trident version

+----------------+----------------+
| SERVER VERSION | CLIENT VERSION |
+----------------+----------------+
| 26.02.0        | 26.02.0        |
+----------------+----------------+
```

Beispielkonfigurationen

Die folgenden Beispiele enthalten Beispielkonfigurationen für die Installation von Trident mit `tridentctl`.

Windows-Knoten

Um Trident auf Windows-Knoten auszuführen:

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

Trennung erzwingen

Weitere Informationen finden Sie unter "Automatisierung des Failovers von zustandsbehafteten Anwendungen mit Trident".

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

Gleichzeitige Trident-Controller-Operationen ermöglichen

Um gleichzeitige Trident-Controller-Operationen für einen verbesserten Durchsatz zu ermöglichen, fügen Sie die `--enable-concurrency` Option während der Installation wie in diesem Beispiel gezeigt hinzu.

> **Tech Preview**: Diese Funktion ist experimentell und unterstützt derzeit eingeschränkte parallele Arbeitsabläufe mit den ONTAP-NAS (nur NFS) und ONTAP-SAN (NVMe für unified ONTAP 9) Treibern, zusätzlich zur bestehenden Tech Preview für den ONTAP-SAN Treiber (iSCSI- und FCP-Protokolle in unified ONTAP 9).

```
tridentctl install --enable-concurrency -n trident
```

tridentctl-Installation anpassen

Sie können das Trident-Installationsprogramm verwenden, um die Installation anzupassen.

Erfahren Sie mehr über das Installationsprogramm

Der Trident-Installer ermöglicht es Ihnen, Attribute anzupassen. Wenn Sie beispielsweise das Trident-Image in ein privates Repository kopiert haben, können Sie den Image-Namen mit `--trident-image` angeben. Wenn Sie sowohl das Trident-Image als auch die benötigten CSI-Sidecar-Images in ein privates Repository kopiert haben, kann es vorzuziehen sein, den Speicherort dieses Repositorys mit dem `--image-registry`-Schalter anzugeben, der die Form `<registry FQDN>[:port]` hat.

> Wenn Sie Trident in einem privaten Repository installieren und den `--image-registry` Schalter verwenden, um den Speicherort des Repositorys anzugeben, verwenden Sie `/netapp/` nicht im Repository-Pfad. Zum Beispiel: `./tridentctl install --image -registry <image-registry> -n <namespace>`

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

Falls Sie die Installation über die Argumente des Installationsprogramms hinaus anpassen müssen, können Sie auch die Bereitstellungsdateien anpassen. Die Verwendung des `--generate-custom-yaml` Parameters erstellt die folgenden YAML-Dateien im `setup` Verzeichnis des Installationsprogramms:

- `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 generiert haben, können Sie sie entsprechend Ihren Bedürfnissen anpassen und dann `--use-custom-yaml` verwenden, um Ihre benutzerdefinierte Bereitstellung zu installieren.

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

Installation mit dem OpenShift zertifizierten Operator

Installieren Sie Trident mit OpenShift OperatorHub

Wenn Sie Red Hat OpenShift verwenden, können Sie NetApp Trident mithilfe des von Red Hat zertifizierten Operators installieren. Verwenden Sie dieses Verfahren, um Trident von der Red Hat OpenShift Container Platform zu installieren.

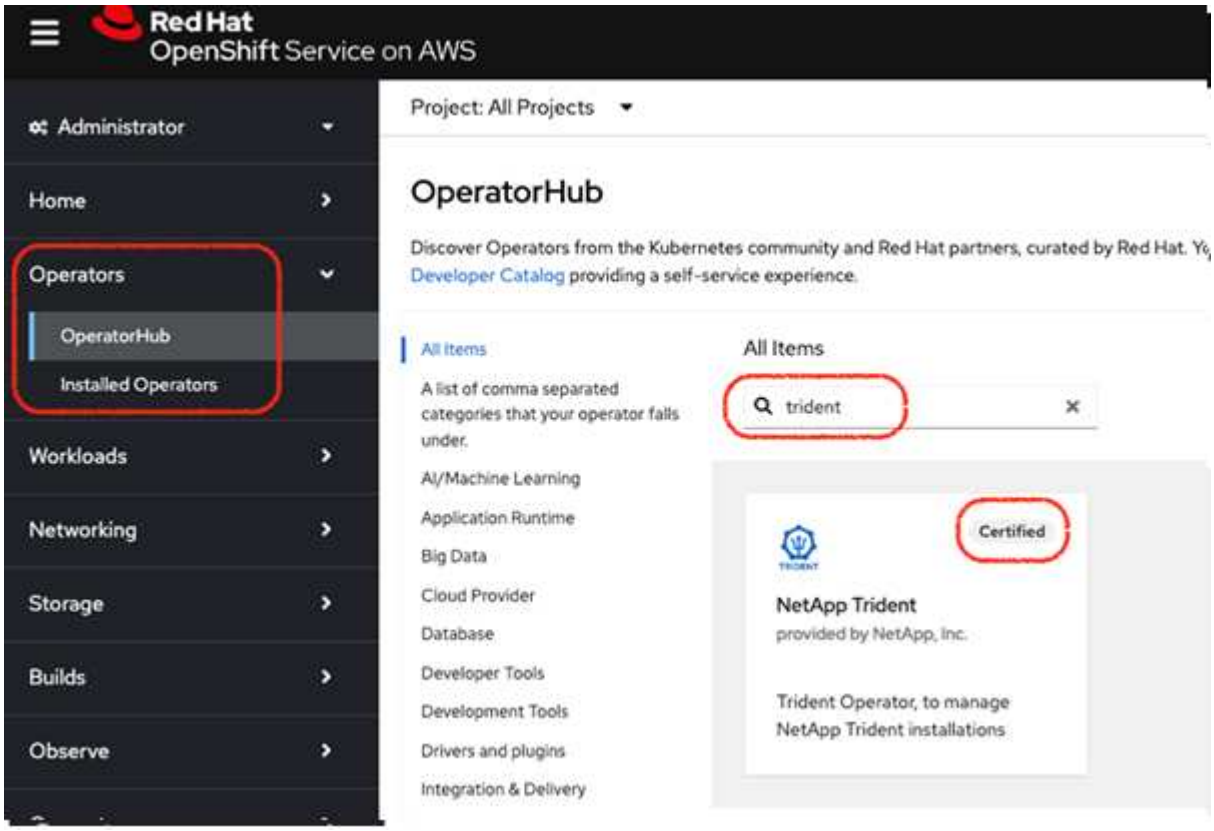
Bevor Sie beginnen

Bevor Sie mit der Installation beginnen, "Bereiten Sie Ihre Umgebung für die Trident Installation vor".

Suchen und installieren Sie den Trident Operator

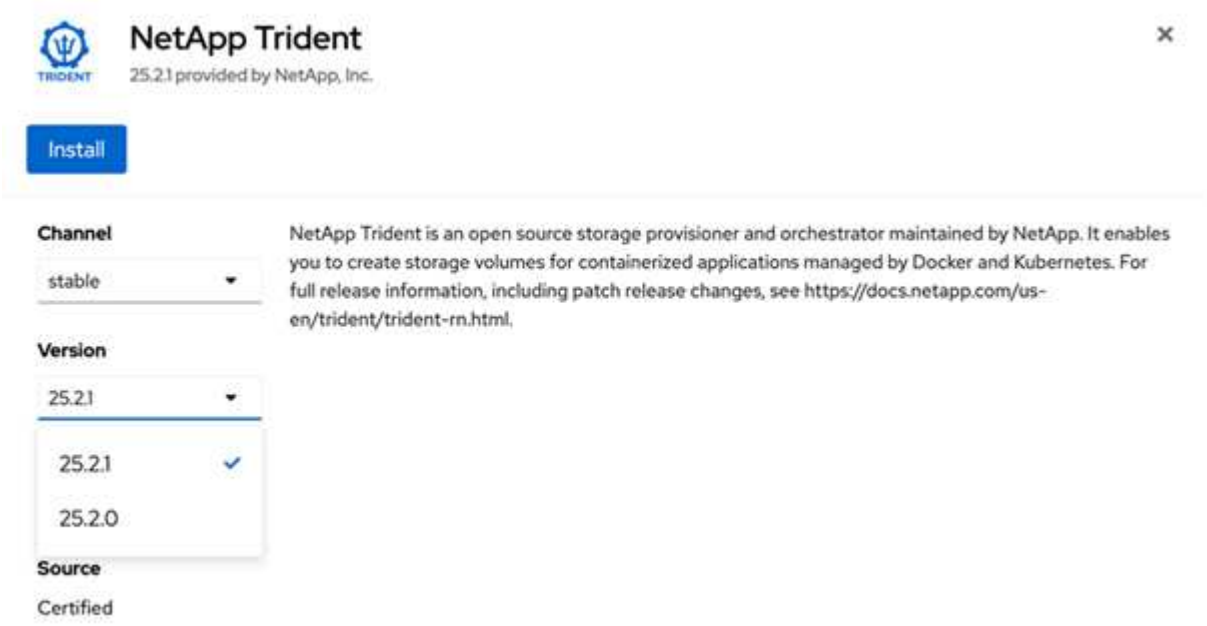
Schritte

1. Navigieren Sie zu OpenShift OperatorHub und suchen Sie nach NetApp Trident.



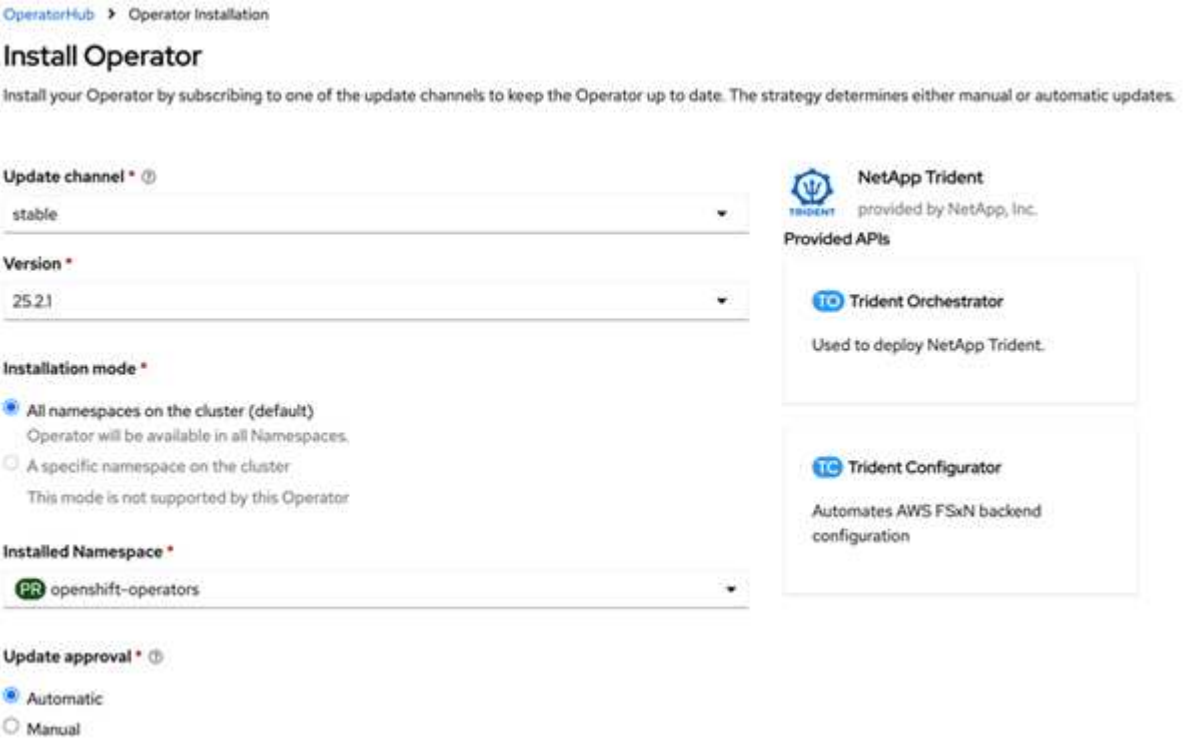
2. Klicken Sie auf **NetApp Trident**, um die Installationseinstellungen zu öffnen.

3. Wählen Sie die gewünschten Optionen aus und klicken Sie auf **Install**, um die Operator-Konfiguration zu öffnen.

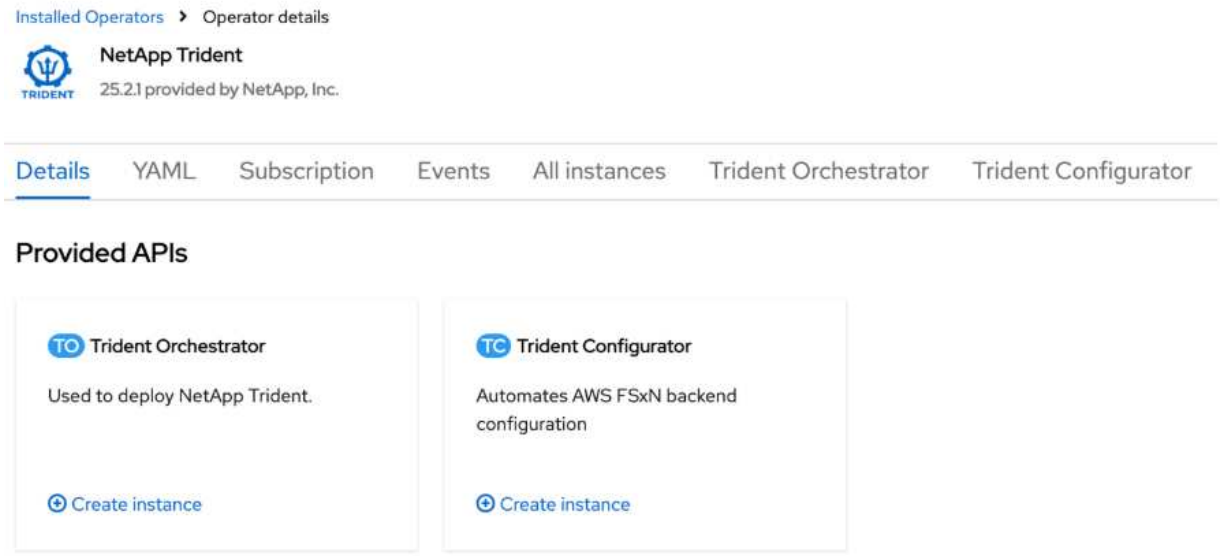


Stellen Sie sicher, dass Sie die aktuellste Operator-Version auswählen.

4. Behalten Sie alle Parameter bei und klicken Sie auf **Install**.



5. Klicken Sie auf **View Operator**, um die Details des Operators anzuzeigen.



6. Klicken Sie auf **YAML view** und fügen Sie Folgendes in das Formular ein:

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
  namespace: openshift-operators
spec:
  IPv6: false
  debug: false
  nodePrep:
  - iscsi
  imageRegistry: ''
  k8sTimeout: 180s
  namespace: trident
  silenceAutosupport: false
```

Die Benutzeroberfläche bietet ein Standardbeispiel. Sie können dieses direkt bearbeiten, anstatt eine vollständige Konfiguration zu kopieren.

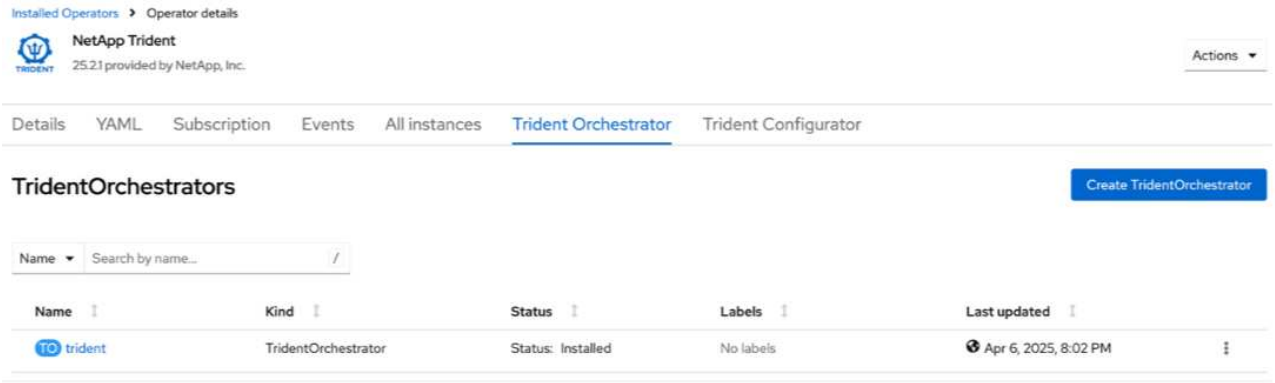
Optional: Gleichzeitigkeit aktivieren

Um Parallelverarbeitung zu ermöglichen, fügen Sie der Spezifikation das folgende Feld hinzu:

```
enableConcurrency: true
```

- Red Hat Enterprise Linux CoreOS (RHCOS) hat iSCSI nicht aktiviert und konfiguriert.
- Sie können den `nodePrep` Parameter hinzufügen, um sowohl iSCSI- als auch Multipath-Dienste auf allen OpenShift Worker-Knoten zu konfigurieren und zu aktivieren.
- Ab OpenShift 4.19 ist die minimale unterstützte Trident-Version für diese Funktion 25.06.1.

1. Klicken Sie auf **Create**; der Trident Orchestrator wird vollständig installiert.



Trident Operator deinstallieren

Schritte

1. Wählen Sie den Trident Operator aus der Liste der installierten Operator aus.
2. Wählen Sie aus, wenn Sie alle Operandeninstanzen aus dem Operator löschen möchten.

   Wenn Sie das Kontrollkästchen **Alle Operandeninstanzen dieses Operators löschen** nicht auswählen, wird Trident nicht deinstalliert.

3. Klicken Sie auf **Uninstall**.

Wechseln Sie zum OpenShift zertifizierten Trident Operator

Sie können vom Community-Operator, einer Helm-basierten Installation oder einem manuell bereitgestellten Operator zum Red Hat OpenShift zertifizierten Trident-Operator wechseln. Der Vorgang umfasst bei jeder Methode die Deinstallation des vorhandenen Operators und anschließend die Installation des zertifizierten Operators mithilfe des OperatorHub.

Bevor Sie beginnen

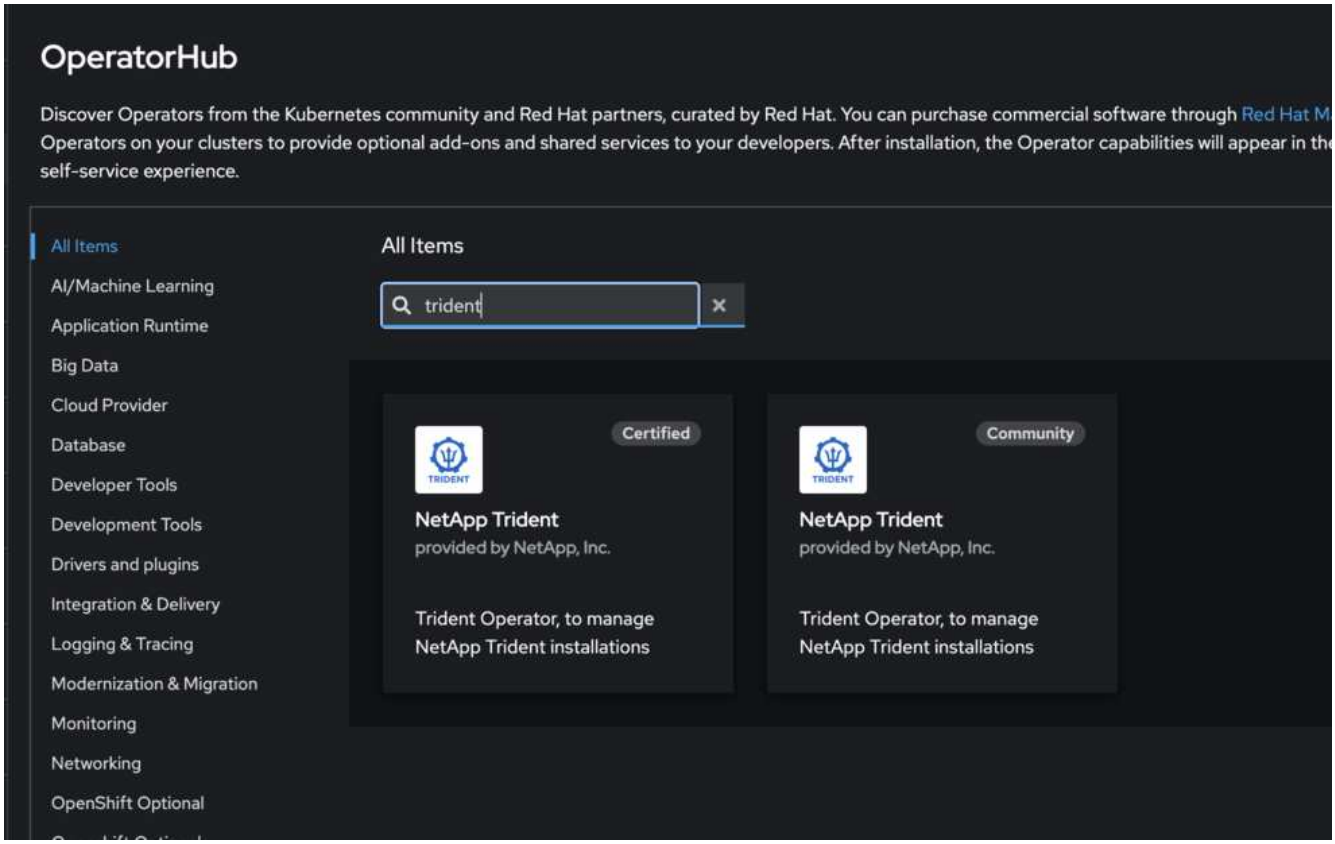
Bevor Sie mit der Installation beginnen, "Bereiten Sie Ihre Umgebung für die Trident Installation vor".

Löschen Sie die `TridentOrchestrator` benutzerdefinierte Ressource (CR) während des Deinstallationsvorgangs nicht. Die `TridentOrchestrator` CR bewahrt Ihre Backend- und Speicherklassenkonfiguration, die nach der Installation des zertifizierten Operators erforderlich ist.

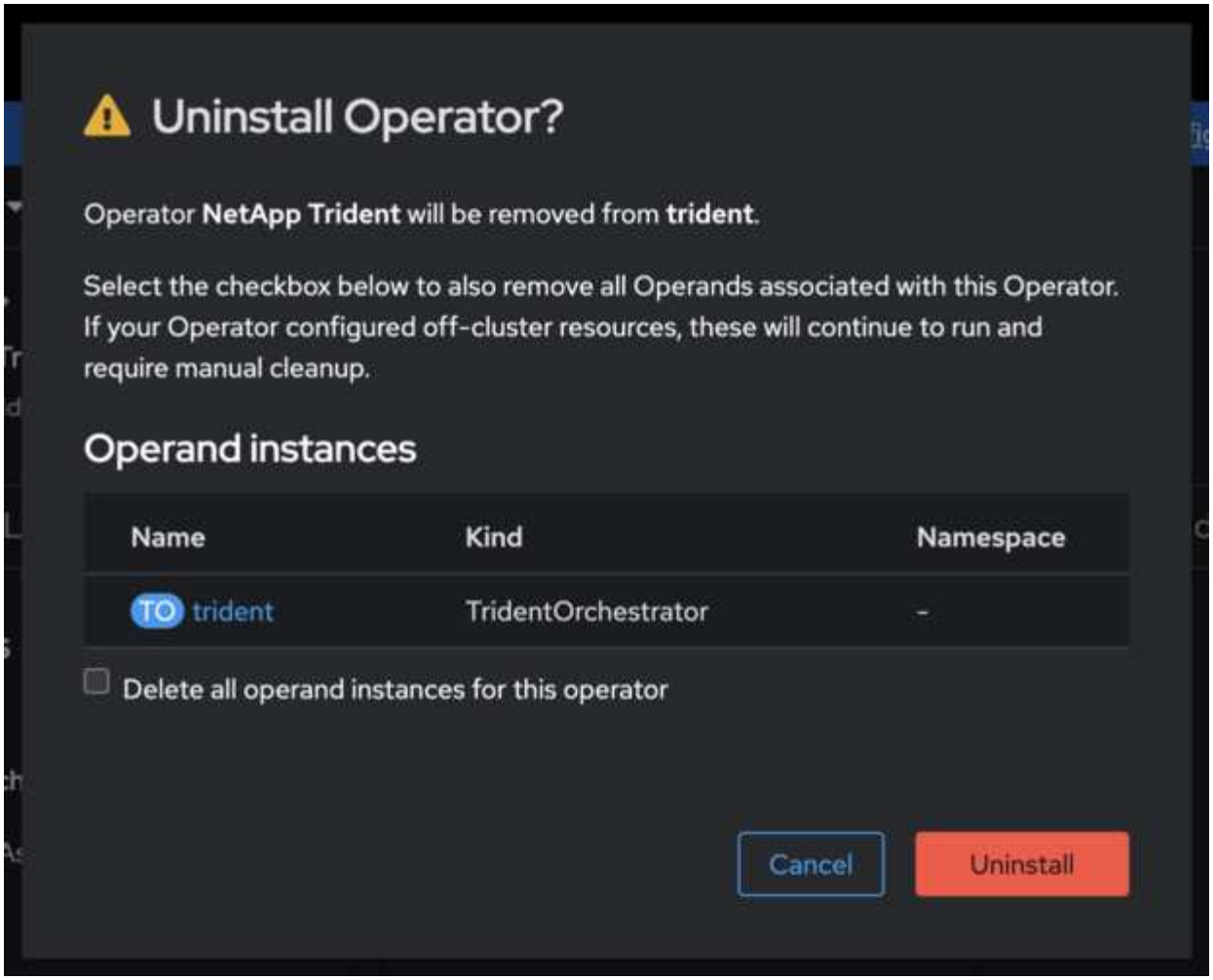
Wechseln Sie vom Community Operator

Schritte

1. Verwenden Sie die OpenShift-Konsole, um zum OperatorHub zu navigieren.



2. Finden Sie den NetApp Trident Community Operator.



Wählen Sie nicht **Alle Operandeninstanzen dieses Operators löschen** aus.

3. Klicken Sie auf **Uninstall**.
4. Nach Abschluss der Deinstallation fahren Sie mit Installieren Sie den OpenShift zertifizierten Operator fort.

Wechsel von einer Helm-basierten Operator-Installation

Schritte

1. Listen Sie das Helm-Release für Ihre Trident Installation auf:

```
helm ls -n trident
```

2. Deinstallieren Sie das Helm-Release:

```
helm uninstall <release-name> -n trident
```

3. Nach Abschluss der Deinstallation fahren Sie mit Installieren Sie den OpenShift zertifizierten Operator fort.

Wechsel von einem manuell bereitgestellten Operator

Wenn Sie Trident durch manuelle Bereitstellung des Operators mithilfe eines `bundle.yaml` aus dem Installationspaket installiert haben, entfernen Sie ihn, indem Sie dasselbe Manifest löschen.

Schritte

1. Löschen Sie die Operator-Bereitstellung mithilfe des Bundle-Manifests:

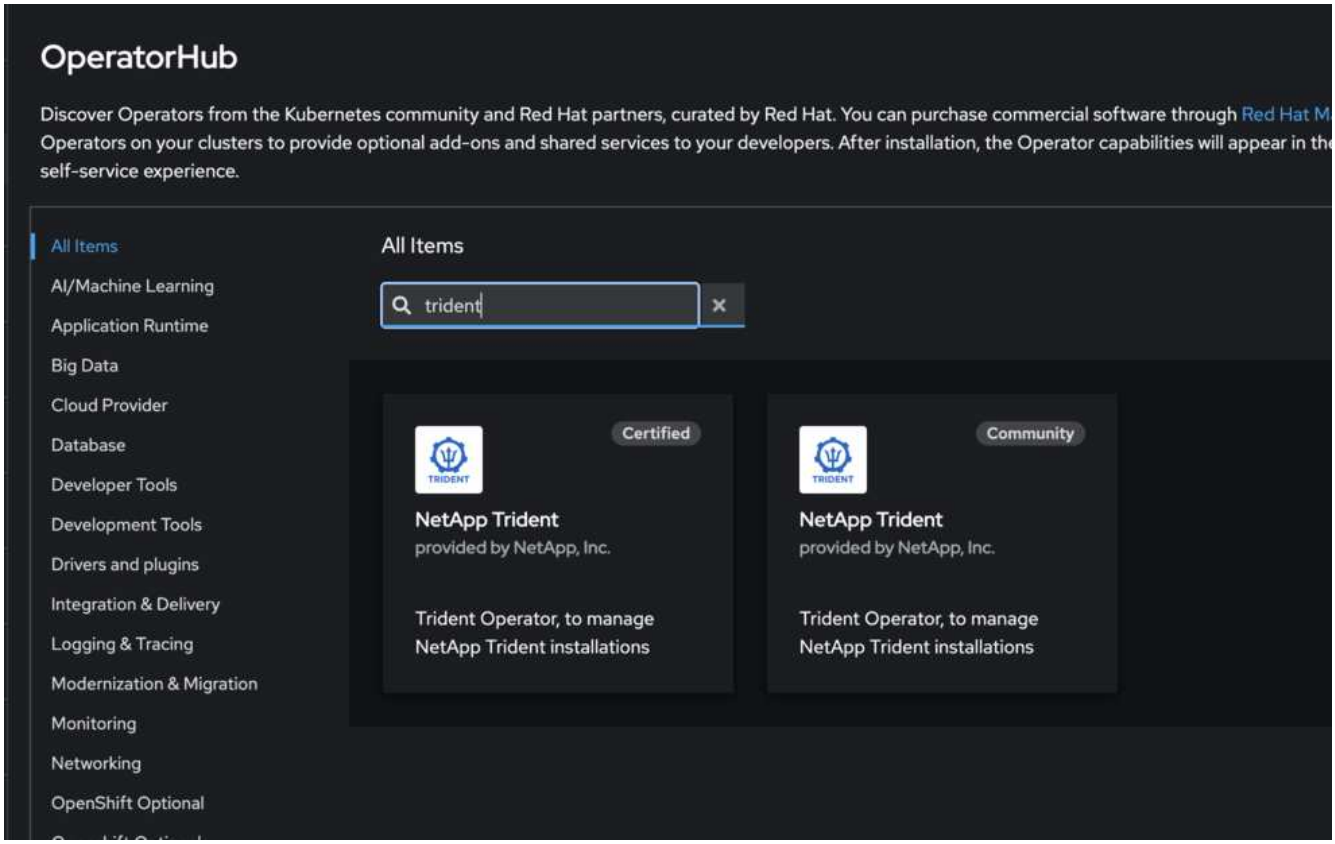
```
kubectl delete -f deploy/bundle.yaml -n trident
```

2. Nach Abschluss der Deinstallation fahren Sie mit Installieren Sie den OpenShift zertifizierten Operator fort.

Installieren Sie den OpenShift zertifizierten Operator

Schritte

1. Navigieren Sie zum Red Hat OperatorHub.
2. Suchen Sie nach dem NetApp Trident Operator und wählen Sie ihn aus.



3. Folgen Sie den Anweisungen auf dem Bildschirm, um den Operator zu installieren.

Überprüfung

- Überprüfen Sie den OperatorHub in der Konsole, um sicherzustellen, dass der neue zertifizierte Operator erfolgreich installiert wurde.

Verwenden Sie Trident

Bereiten Sie den Worker-Knoten vor

Alle Worker-Knoten im Kubernetes-Cluster müssen die für Ihre Pods bereitgestellten Volumes einbinden können. Zur Vorbereitung der Worker-Knoten müssen Sie je nach Treiberauswahl NFS-, iSCSI-, NVMe/TCP- oder FC-Tools installieren.

Die richtigen Werkzeuge auswählen

Wenn Sie eine Kombination von Treibern verwenden, sollten Sie alle erforderlichen Tools für Ihre Treiber installieren. Neuere Versionen von Red Hat Enterprise Linux CoreOS (RHCOS) haben die Tools standardmäßig installiert.

NFS-Tools

"Installieren Sie die NFS-Tools"Wenn Sie verwenden: `ontap-nas`, `ontap-nas-economy`, `ontap-nas-flexgroup` oder `azure-netapp-files`.

iSCSI-Tools

"Installieren Sie die iSCSI-Tools"Wenn Sie `ontap-san`, `ontap-san-economy`, `solidfire-san` verwenden.

NVMe-Tools

"Installieren Sie die NVMe-Tools" wenn Sie `ontap-san` für Nonvolatile Memory Express (NVMe) über TCP (NVMe/TCP)-Protokoll verwenden.

> NetApp empfiehlt ONTAP 9.12 oder höher für NVMe/TCP.

SCSI über FC-Tools

Weitere Informationen zur Konfiguration Ihrer FC- und FC-NVMe-SAN-Hosts finden Sie unter "Möglichkeiten zur Konfiguration von FC- FC-NVMe SAN-Hosts".

"Installieren Sie die FC-Tools" wenn Sie `ontap-san` mit sanType `fcp` (SCSI über FC) verwenden.

Zu beachtende Punkte: * SCSI über FC wird in OpenShift- und KubeVirt-Umgebungen unterstützt. * SCSI über FC wird auf Docker nicht unterstützt. * Die iSCSI-Selbstreparatur ist bei SCSI über FC nicht anwendbar.

SMB-Tools

"Bereiten Sie die Bereitstellung von SMB-Volumes vor" wenn Sie `ontap-nas` verwenden, um SMB-Volumes bereitzustellen.

Knotendiensterkennung

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

> Die Knotendiensterkennung identifiziert gefundene Dienste, garantiert jedoch nicht, dass die Dienste ordnungsgemäß konfiguriert sind. Umgekehrt garantiert das Fehlen eines gefundenen Dienstes nicht, dass die Volume-Einbindung fehlschlägt.

Ereignisse überprüfen

Trident erstellt Ereignisse für den Knoten, um die gefundenen Dienste zu identifizieren. Um diese Ereignisse anzuzeigen, führen Sie aus:

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

Entdeckte Dienste überprüfen

Trident identifiziert die für jeden Knoten auf dem Trident node CR aktivierten Dienste. Um die ermittelten Dienste anzuzeigen, führen Sie aus:

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

NFS-Volumes

Installieren Sie die NFS-Tools mithilfe der Befehle für Ihr Betriebssystem. Stellen Sie sicher, dass der NFS-Service während der Boot-Zeit gestartet wird.

RHEL 8+

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

Ubuntu

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

Starten Sie Ihre Worker-Knoten nach der Installation der NFS-Tools neu, um Fehler beim Anhängen von Volumes an Container zu vermeiden.

iSCSI-Volumes

Trident kann automatisch eine iSCSI-Sitzung herstellen, LUNs scannen und Multipath-Geräte erkennen, sie formatieren und in einen Pod einbinden.

iSCSI Selbstreparaturfunktionen

Für ONTAP-Systeme führt Trident alle fünf Minuten eine iSCSI-Selbstreparatur durch, um:

1. **Identifizieren** Sie den gewünschten iSCSI-Sitzungsstatus und den aktuellen iSCSI-Sitzungsstatus.
2. **Vergleichen** Sie den Sollzustand mit dem Istzustand, um notwendige Reparaturen zu ermitteln. Trident legt die Reparaturprioritäten fest und entscheidet, wann Reparaturen vorgezogen werden sollen.
3. **Führen Sie Reparaturen durch**, die erforderlich sind, um den aktuellen iSCSI-Sitzungsstatus in den gewünschten iSCSI-Sitzungsstatus zurückzusetzen.

Protokolle der Selbstreparaturaktivität befinden sich im `trident-main` Container auf dem jeweiligen Daemonset-Pod. Um Protokolle anzuzeigen, müssen Sie während der Trident-Installation `debug` auf „true“ gesetzt haben.

Die Selbstreparaturfunktionen von Trident iSCSI können dazu beitragen, Folgendes zu verhindern:

- Veraltete oder fehlerhafte iSCSI-Sitzungen, die nach einem Netzwerkverbindungsproblem auftreten können. Im Falle einer veralteten Sitzung wartet Trident sieben Minuten, bevor es sich ausloggt, um die Verbindung mit einem Portal wiederherzustellen.

  Wenn beispielsweise CHAP-Geheimnisse auf dem Speicherkontroller rotiert wurden und die Netzwerkverbindung abbricht, können die alten (*veralteten*) CHAP-Geheimnisse erhalten bleiben. Selbstreparatur kann dies erkennen und die Sitzung automatisch wiederherstellen, um die aktualisierten CHAP-Geheimnisse anzuwenden.

- Fehlende iSCSI-Sitzungen
- Fehlende LUNs

Punkte, die Sie vor einem Upgrade von Trident beachten sollten

- Wenn nur pro-Knoten-igroups (eingeführt in 23.04+) verwendet werden, wird die iSCSI-Selbstreparatur SCSI-Neuscans für alle Geräte im SCSI-Bus initiieren.
- Wenn nur Backend-bezogene igroups (veraltet ab 23.04) verwendet werden, wird die iSCSI-Selbstreparatur SCSI-Scans für exakte LUN-IDs im SCSI-Bus initiieren.
- Wenn eine Mischung aus knotenbezogenen igroups und backendbezogenen igroups verwendet wird, initiiert die iSCSI-Selbstreparatur SCSI-Rescans für exakte LUN-IDs im SCSI-Bus.

Installieren Sie die iSCSI-Tools

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

Bevor Sie beginnen

- Jeder Knoten im Kubernetes-Cluster muss einen eindeutigen IQN besitzen. **Dies ist eine notwendige Voraussetzung**.
- Wenn Sie RHCOS Version 4.5 oder höher oder eine andere RHEL-kompatible Linux-Distribution mit dem `solidfire-san` Treiber und Element OS 12.5 oder früher verwenden, stellen Sie sicher, dass der CHAP-Authentifizierungsalgorithmus in `/etc/iscsi/iscsid.conf` auf MD5 eingestellt ist. Sichere, FIPS-konforme CHAP-Algorithmen SHA1, SHA-256 und SHA3-256 sind mit Element 12.7 verfügbar.

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

- Bei Verwendung von Worker-Knoten, auf denen RHEL/Red Hat Enterprise Linux CoreOS (RHCOS) mit iSCSI-PVs ausgeführt wird, geben Sie die `discard` mountOption in der StorageClass an, um eine Inline-Speicherplatzfreigabe durchzuführen. Siehe "Red Hat Dokumentation".
- Stellen Sie sicher, dass Sie auf die neueste Version des `multipath-tools` aktualisiert haben.

RHEL 8+

1. Installieren Sie die folgenden Systempakete:

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

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

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

3. Scanvorgang 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
```

> Stellen Sie sicher, dass `/etc/multipath.conf` `find_multipaths no` unter `defaults` enthalten ist.

5. Stellen Sie sicher, dass `iscsid` und `multipathd` ausgeführt werden:

```
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. Prüfen Sie, ob die open-iscsi-Version 2.0.874-5ubuntu2.10 oder höher (für bionic) oder 2.0.874-7.1ubuntu6.1 oder höher (für focal) ist:

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

3. Scanvorgang 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
```

> Stellen Sie sicher, dass `/etc/multipath.conf` `find_multipaths no` unter `defaults` enthalten ist.

5. Stellen Sie sicher, dass `open-iscsi` und `multipath-tools` aktiviert sind und ausgeführt werden:

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

> Für Ubuntu 18.04 müssen Sie die Zielports mit `iscsiadm` entdecken, bevor Sie `open-iscsi` starten, damit der iSCSI-Daemon gestartet werden kann. Alternativ können Sie den `iscsi` Dienst so ändern, dass er `iscsid` automatisch startet.

iSCSI-Selbstreparatur konfigurieren oder deaktivieren

Sie können die folgenden Trident iSCSI Selbstreparatur-Einstellungen konfigurieren, um veraltete Sitzungen zu beheben:

- **iSCSI Selbstreparaturintervall**: Legt die Häufigkeit fest, mit der die iSCSI Selbstreparatur aufgerufen wird (Standard: 5 Minuten). Sie können es so konfigurieren, dass es häufiger durch die Angabe einer kleineren Zahl oder seltener durch die Angabe einer größeren Zahl ausgeführt wird.

Durch das Setzen des iSCSI-Selbstreparaturintervalls auf 0 wird die iSCSI-Selbstreparatur vollständig gestoppt. Wir empfehlen nicht, die iSCSI-Selbstreparatur zu deaktivieren; sie sollte nur in bestimmten Szenarien deaktiviert werden, wenn die iSCSI-Selbstreparatur nicht wie vorgesehen funktioniert oder zu Debugging-Zwecken.

- **iSCSI-Selbstreparatur-Wartezeit**: Bestimmt die Dauer, die die iSCSI-Selbstreparatur wartet, bevor eine fehlerhafte Sitzung ausgeloggt und ein erneuter Anmeldeversuch unternommen wird (Standard: 7 Minuten). Sie können einen höheren Wert einstellen, sodass Sitzungen, die als fehlerhaft erkannt werden, länger warten müssen, bevor sie ausgeloggt werden und dann ein erneuter Anmeldeversuch unternommen wird, oder einen niedrigeren Wert, um früher auszuloggen und sich erneut anzumelden.

Helm

Um die Einstellungen für die iSCSI-Selbstreparatur zu konfigurieren oder zu ändern, übergeben Sie die `iscsiSelfHealingInterval` und `iscsiSelfHealingWaitTime` Parameter während der Helm-Installation oder des Helm-Updates.

Im folgenden Beispiel wird das iSCSI Selbstreparaturintervall auf 3 Minuten und die Selbstreparatur-Wartezeit auf 6 Minuten eingestellt:

```
helm install trident trident-operator-100.2602.0.tgz --set
iscsiSelfHealingInterval=3m0s --set iscsiSelfHealingWaitTime=6m0s -n
trident
```

tridentctl

Um die Einstellungen für die iSCSI-Selbstreparatur zu konfigurieren oder zu ändern, übergeben Sie die `iscsi-self-healing-interval` und `iscsi-self-healing-wait-time` Parameter während der Installation oder Aktualisierung von tridentctl.

Im folgenden Beispiel wird das iSCSI Selbstreparaturintervall auf 3 Minuten und die Selbstreparatur-Wartezeit auf 6 Minuten eingestellt:

```
tridentctl install --iscsi-self-healing-interval=3m0s --iscsi-self
-healing-wait-time=6m0s -n trident
```

NVMe/TCP Volumes

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

- NVMe erfordert RHEL 9 oder neuer.
- Falls die Kernelversion Ihres Kubernetes-Knotens zu alt ist oder das NVMe-Paket für Ihre Kernelversion nicht verfügbar ist, müssen Sie möglicherweise die Kernelversion Ihres Knotens auf eine Version mit dem NVMe-Paket aktualisieren.

RHEL 9

```
sudo yum install nvme-cli
sudo yum install linux-modules-extra-$(uname -r)
sudo modprobe nvme-tcp
```

Ubuntu

```
sudo apt install nvme-cli
sudo apt -y install linux-modules-extra-$(uname -r)
sudo modprobe nvme-tcp
```

Installation überprüfen

Überprüfen Sie nach der Installation, ob jeder Knoten im Kubernetes-Cluster einen eindeutigen NQN besitzt, indem Sie den folgenden Befehl verwenden:

```
cat /etc/nvme/hostnqn
```

Trident ändert den `ctrl_device_tmo` Wert, um sicherzustellen, dass NVMe den Pfad nicht aufgibt, falls er ausfällt. Ändern Sie diese Einstellung nicht.

SCSI über FC-Volumes

Sie können jetzt das Fibre Channel (FC)-Protokoll mit Trident verwenden, um Speicherressourcen auf dem ONTAP-System bereitzustellen und zu verwalten.

Voraussetzungen

Konfigurieren Sie die erforderlichen Netzwerk- und Knoteneinstellungen für FC.

Netzwerkeinstellungen

1. Ermitteln Sie die WWPN der Zielschnittstellen. Weitere Informationen sind unter "Netzwerkschnittstelle anzeigen" verfügbar.
2. Ermitteln Sie die WWPN für die Schnittstellen auf dem Initiator (Host).

   Siehe die entsprechenden Dienstprogramme des Host-Betriebssystems.

3. Konfigurieren Sie das Zoning auf dem FC-Switch mithilfe der WWPNs des Hosts und des Ziels.

   Informationen finden Sie in der jeweiligen Switch-Anbieter-Dokumentation.

   Weitere Einzelheiten entnehmen Sie bitte der folgenden ONTAP-Dokumentation:

   - "Fibre Channel und FCoE-Zoning-Übersicht"

- "Möglichkeiten zur Konfiguration von FC- FC-NVMe SAN-Hosts"

Installieren Sie die FC-Tools

Installieren Sie die FC-Tools mithilfe der Befehle für Ihr Betriebssystem.

- Bei Verwendung von Worker-Knoten, auf denen RHEL/Red Hat Enterprise Linux CoreOS (RHCOS) mit FC-PVs ausgeführt werden, geben Sie die `discard` mountOption in der StorageClass an, um die Inline-Speicherplatzfreigabe durchzuführen. Siehe "Red Hat Dokumentation".

RHEL 8+

1. Installieren Sie die folgenden Systempakete:

```
sudo yum install -y lsscsi device-mapper-multipath
```

2. Multipathing aktivieren:

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

> Stellen Sie sicher, dass `/etc/multipath.conf` `find_multipaths no` unter `defaults` enthalten ist.

3. Stellen Sie sicher, dass `multipathd` läuft:

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

Ubuntu

1. Installieren Sie die folgenden Systempakete:

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

2. 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
```

> Stellen Sie sicher, dass `/etc/multipath.conf` `find_multipaths no` unter `defaults` enthalten ist.

3. Stellen Sie sicher, dass `multipath-tools` aktiviert ist und ausgeführt wird:

```
sudo systemctl status multipath-tools
```

Bereiten Sie die Bereitstellung von SMB-Volumes vor

Sie können SMB-Volumes mit `ontap-nas` Treibern bereitstellen.

> Sie müssen sowohl das NFS- als auch das SMB/CIFS-Protokoll auf der SVM konfigurieren, um ein `ontap-nas-economy` SMB-Volume für ONTAP On-Premises-Cluster zu erstellen. Wenn eines dieser Protokolle nicht konfiguriert ist, schlägt die Erstellung des SMB-Volumes fehl.

> `autoExportPolicy` is not supported for SMB-Volumes.

Bevor Sie beginnen

Bevor Sie SMB-Volumes bereitstellen können, müssen Sie Folgendes haben.

- Ein Kubernetes-Cluster mit einem Linux-Controller-Knoten und mindestens einem Windows-Worker-Knoten, auf dem Windows Server 2022 läuft. Trident unterstützt SMB-Volumes nur, wenn sie in Pods eingebunden sind, die auf Windows-Knoten ausgeführt werden.
- Mindestens ein Trident secret, das Ihre Active Directory-Anmeldeinformationen enthält. Um ein secret zu generieren `smbcreds`:

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

- Ein als Windows-Dienst konfigurierter CSI-Proxy. Um einen `csi-proxy` zu konfigurieren, siehe "GitHub: CSI Proxy" oder "GitHub: CSI-Proxy für Windows" für Kubernetes-Knoten, die unter Windows laufen.

Schritte

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

   > SMB-Shares sind für Amazon FSx for ONTAP erforderlich.

   Sie können die SMB-Administratorfreigaben auf zwei Arten erstellen: entweder mit dem "Microsoft Management Console" Shared Folders Snap-In oder mit der ONTAP CLI. Um die SMB-Freigaben mit der ONTAP CLI zu erstellen:

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

      Der `vserver cifs share create` Befehl prüft den in der Option -path beim Erstellen der Freigabe angegebenen Pfad. Wenn der angegebene Pfad nicht existiert, schlägt der Befehl fehl.

   b. Erstellen Sie eine SMB-Freigabe, die dem angegebenen SVM zugeordnet ist:

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

   c. Überprüfen Sie, ob die Freigabe erstellt wurde:

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

Weitere Einzelheiten finden Sie in "Erstellen Sie eine SMB-Freigabe".

2. Bei der Erstellung des Backends müssen Sie Folgendes konfigurieren, um SMB-Volumes anzugeben. Alle FSx for ONTAP Backend-Konfigurationsoptionen finden Sie unter "FSx for ONTAP Konfigurationsoptionen und Beispiele".

Parameter	Beschreibung	Beispiel
`smbShare`	Sie können eines der folgenden angeben: den Namen einer SMB-Freigabe, die mit der Microsoft Management Console oder ONTAP CLI erstellt wurde; einen Namen, damit Trident die SMB-Freigabe erstellen kann; oder Sie können den Parameter leer lassen, um den gemeinsamen Freigabezugriff auf Volumes zu verhindern. Dieser Parameter ist optional für lokale ONTAP. Dieser Parameter ist für Amazon FSx for ONTAP Backends erforderlich und darf nicht leer sein.	`smb-share`
`nasType`	**Muss auf `smb` gesetzt werden.** Wenn null, wird standardmäßig `nfs` verwendet.	`smb`
`securityStyle`	Sicherheitsstil für neue Volumes. **Muss auf `ntfs` oder `mixed` für SMB-Volumes eingestellt werden.**	`ntfs` or `mixed` für SMB-Volumes
`unixPermissions`	Modus für neue Volumes. **Muss für SMB-Volumes leer bleiben.**	""

Backends konfigurieren und verwalten

Backends konfigurieren

Ein Backend definiert die Beziehung zwischen Trident und einem Speichersystem. Es gibt Trident vor, wie Trident mit diesem Speichersystem kommuniziert und wie Trident Volumes daraus bereitstellen soll.

Trident bietet automatisch Speicherpools von Backends an, die den Anforderungen entsprechen, die durch eine Speicherklasse definiert sind. Erfahren Sie, wie Sie das Backend für Ihr Speichersystem konfigurieren.

- "Konfigurieren eines Azure NetApp Files-Backends"
- "Konfigurieren eines Google Cloud NetApp Volumes-Backends"
- "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"
- "Verwenden Sie Trident mit Amazon FSx for NetApp ONTAP"

Azure NetApp Files

Konfigurieren eines Azure NetApp Files-Backends

Verwenden Sie Azure NetApp Files als Backend für Trident. Dieses Backend unterstützt NFS- und SMB-Volumes. Trident unterstützt verwaltete Identitäten und Workload-Identität für Azure Kubernetes Service (AKS)-Cluster.

Unterstützte Azure Cloud-Umgebungen

Trident unterstützt Azure NetApp Files-Backends in mehreren Azure Cloud-Umgebungen.

Unterstützte Azure-Clouds umfassen:

- Azure Commercial
- Azure Government (Azure Government / MAG)

Wenn Sie Trident bereitstellen oder ein Azure NetApp Files-Backend konfigurieren, stellen Sie sicher, dass Azure Resource Manager und Authentifizierung-Endpunkte mit Ihrer Azure-Cloudumgebung übereinstimmen.

Überprüfen Sie die Unterstützung des Azure NetApp Files-Treibers

Trident stellt den folgenden Azure NetApp Files-Speichertreiber bereit.

Unterstützte Zugriffsmodi sind *ReadWriteOnce* (RWO), *ReadOnlyMany* (ROX), *ReadWriteMany* (RWX) und *ReadWriteOncePod* (RWOP).

Treiber	Protokoll	volumeMode	Unterstützte Zugriffsmodi	Unterstützte Dateisysteme
`azure-netapp-files`	NFS SMB	Dateisystem	RWO, ROX, RWX, RWOP	`nfs`, `smb`

Überlegungen zur Überprüfung

- Azure NetApp Files unterstützt keine Volumes unter 50 GiB. Trident erstellt ein 50-GiB-Volume, wenn ein kleineres Volume angefordert wird.
- Trident unterstützt SMB-Volumes nur, wenn sie in Pods eingebunden sind, die auf Windows-Knoten ausgeführt werden.
- Azure NetApp Files-Bereitstellungen in nicht-kommerziellen Azure-Clouds erfordern cloud-spezifische Azure Resource Manager- und Authentifizierungsendpunkte. Stellen Sie sicher, dass Trident und jede Backend-Konfiguration die für Ihre Azure-Cloud-Umgebung geeigneten Endpunkte verwenden.

Verwenden Sie verwaltete Identitäten für AKS

Trident unterstützt "verwaltete Identitäten" für AKS-Cluster.

Wenn Sie `tridentctl` verwenden, um Azure NetApp Files Backends zu erstellen oder zu verwalten, stellen Sie sicher, dass es für die richtige Azure-Cloudumgebung konfiguriert ist.

Um verwaltete Identitäten zu verwenden, müssen Sie Folgendes haben:

- Ein mit AKS bereitgestellter Kubernetes-Cluster

- Auf dem AKS Kubernetes-Cluster konfigurierte verwaltete Identitäten
- Trident installiert mit `cloudProvider` auf `"Azure"`

Trident Operator

Bearbeiten `tridentorchestrator_cr.yaml` und setzen `cloudProvider` auf `"Azure"`.

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  imagePullPolicy: IfNotPresent
  cloudProvider: "Azure"
```

Helm

Das folgende Beispiel installiert Trident und setzt `cloudProvider` mithilfe der Umgebungsvariable `$CP`:

```
helm install trident trident-operator-100.2602.0.tgz --create-namespace
--namespace <trident-namespace> --set cloudProvider=$CP
```

<code>tridentctl</code>

Das folgende Beispiel installiert Trident und setzt die `cloud-provider`-Flag auf `Azure`:

```
tridentctl install --cloud-provider="Azure" -n trident
```

Workload-Identität für AKS verwenden

Die Workload-Identität ermöglicht es Kubernetes-Pods, auf Azure-Ressourcen zuzugreifen, indem sie sich als Workload-Identität authentifizieren.

Wenn Sie `tridentctl` verwenden, um Azure NetApp Files Backends zu erstellen oder zu verwalten, stellen Sie sicher, dass es für die richtige Azure-Cloudumgebung konfiguriert ist.

Um die Workload-Identität zu verwenden, müssen Sie Folgendes haben:

- Ein mit AKS bereitgestellter Kubernetes-Cluster
- Workload-Identität und oidc-issuer auf dem AKS Kubernetes-Cluster konfiguriert
- Trident wurde mit `cloudProvider` auf `"Azure"` gesetzt und `cloudIdentity` auf den Wert der Workload-Identität installiert

Trident Operator

Bearbeiten `tridentorchestrator_cr.yaml` und setzen `cloudProvider` auf `"Azure"`. Setzen `cloudIdentity` auf `azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxx`.

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  imagePullPolicy: IfNotPresent
  cloudProvider: "Azure"
  cloudIdentity: 'azure.workload.identity/client-id: xxxxxxxx-xxxx-
xxxx-xxxx-xxxxxxxxxxx' # Edit
```

Helm

Legen Sie die Werte für die Flags **cloud-provider (CP)** und **cloud-identity (CI)** mithilfe der folgenden Umgebungsvariablen fest:

```
export CP="Azure"
export CI="'azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxxx'"
```

Das folgende Beispiel installiert Trident und setzt `cloudProvider` mit `$CP` und setzt `cloudIdentity` mit `$CI`:

```
helm install trident trident-operator-100.6.0.tgz --set
cloudProvider=$CP --set cloudIdentity="$CI"
```

<code>tridentctl</code>

Legen Sie die Werte für die Flags **cloud provider** und **cloud identity** mithilfe der folgenden Umgebungsvariablen fest:

```
export CP="Azure"
export CI="azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxxx"
```

Das folgende Beispiel installiert Trident und setzt `cloud-provider` auf `$CP` und `cloud-identity` auf `$CI`:

```
tridentctl install --cloud-provider=$CP --cloud-identity="$CI" -n
trident
```

Bereiten Sie die Konfiguration eines Azure NetApp Files-Backends vor

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

Unterstützte Azure Cloud-Umgebungen

Trident unterstützt Azure NetApp Files-Backends in mehreren Azure Cloud-Umgebungen.

Unterstützte Azure-Clouds umfassen:

- Azure Commercial
- Azure Government (Azure Government / MAG)

Stellen Sie bei der Vorbereitung Ihrer Umgebung sicher, dass Ihr Azure-Abonnement, Ihre Identitätskonfiguration und Ihre Azure NetApp Files-Ressourcen in der entsprechenden Azure-Cloudumgebung erstellt werden.

Voraussetzungen für NFS- und SMB-Volumes

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

Um ein "Azure NetApp Files" Backend zu konfigurieren und zu verwenden, benötigen Sie Folgendes:

> - `subscriptionID`, `tenantID`, `clientID`, `location` und `clientSecret` sind optional, wenn verwaltete Identitäten auf einem AKS-Cluster verwendet werden.
> - `tenantID`, `clientID` und `clientSecret` sind optional, wenn eine Cloud-Identität auf einem AKS-Cluster verwendet wird.
> - Azure NetApp Files-Bereitstellungen in nicht-kommerziellen Azure-Clouds erfordern cloud-spezifische Azure Resource Manager- und Authentifizierungsendpunkte. Stellen Sie sicher, dass Trident und jede Backend-Konfiguration die für Ihre Azure-Cloud-Umgebung geeigneten Endpunkte verwenden.

- Ein Kapazitätspool. Siehe "Microsoft: Erstellen eines Kapazitätspools für Azure NetApp Files".
- Ein an Azure NetApp Files delegiertes Subnetz. Siehe "Microsoft: Delegieren Sie ein Subnetz an Azure NetApp Files".
- `subscriptionID` aus einem Azure-Abonnement mit aktivierten 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 entweder Folgendes verwenden:
  - Die Rolle Eigentümer oder Mitwirkender "von Azure vordefiniert".
  - A "benutzerdefinierte Contributor-Rolle" auf Abonnementebene (`assignableScopes`) mit den folgenden Berechtigungen, die auf nur das beschränkt sind, was Trident benötigt. Nach dem Erstellen der benutzerdefinierten Rolle "Weisen Sie die Rolle mithilfe des Azure-Portals zu".

Benutzerdefinierte Contributor-Rolle

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

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/read",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/write",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/delete",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/snapshots/
read",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/snapshots/
write",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/snapshots/
delete",

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

"Microsoft.Features/featureProviders/subscriptionFeatureRegistrat
ions/read",

"Microsoft.Features/featureProviders/subscriptionFeatureRegistrat
ions/write",

"Microsoft.Features/featureProviders/subscriptionFeatureRegistrat
ions/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": []
        }
      ]
    }
}
```

- Die Azure `location`, die mindestens einen "delegiertes Subnetz"enthält. Ab Trident 22.01 ist der `location`-Parameter ein Pflichtfeld auf oberster Ebene der Backend-Konfigurationsdatei. In virtuellen Pools angegebene Standortwerte werden ignoriert.
- Um `Cloud Identity` zu verwenden, holen Sie die `client ID` von einem "vom Benutzer zugewiesene verwaltete Identität" und geben Sie diese ID in `azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxx` an.

Zusätzliche Anforderungen für SMB-Volumina

Zum Erstellen eines SMB-Volumes müssen Sie Folgendes haben:

- Active Directory ist konfiguriert und mit Azure NetApp Files verbunden. Siehe "Microsoft: Active Directory-Verbindungen für Azure NetApp Files erstellen und verwalten".
- Ein Kubernetes-Cluster mit einem Linux-Controller-Knoten und mindestens einem Windows-Worker-Knoten, auf dem Windows Server 2022 läuft. Trident unterstützt SMB-Volumes nur, wenn sie in Pods eingebunden sind, die auf Windows-Knoten ausgeführt werden.
- Mindestens ein Trident-Geheimnis, das Ihre Active Directory-Anmeldeinformationen enthält, damit Azure NetApp Files sich bei Active Directory authentifizieren kann. Um ein secret zu generieren `smbcreds`:

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

- Ein als Windows-Dienst konfigurierter CSI-Proxy. Um einen `csi-proxy` zu konfigurieren, siehe "GitHub: CSI Proxy" oder "GitHub: CSI-Proxy für Windows" für Kubernetes-Knoten, die unter Windows laufen.

Azure NetApp Files Backend-Konfigurationsoptionen und Beispiele

Erfahren Sie mehr über die NFS- und SMB-Backend-Konfigurationsoptionen für Azure

NetApp Files und sehen Sie sich Konfigurationsbeispiele an.

Backend-Konfigurationsoptionen

Trident verwendet Ihre Backend-Konfiguration (Subnetz, virtuelles Netzwerk, Servicelevel und Standort), um Azure NetApp Files Volumes auf Kapazitätspools zu erstellen, die am angeforderten Standort verfügbar sind und dem angeforderten Servicelevel und Subnetz entsprechen.

Azure NetApp Files-Backends bieten diese Konfigurationsoptionen.

Parameter	Beschreibung	Standard
`version`	Backend-Konfigurationsversion.	Immer 1
`storageDriverName`	Name des Speichertreibers	"azure-netapp-files"
`backendName`	Benutzerdefinierter Name für das Speicher-Backend	Driver name + "_" + zufällige Zeichen
`subscriptionID`	Die Abonnement-ID Ihres Azure-Abonnements Optional, wenn verwaltete Identitäten in einem AKS-Cluster aktiviert sind.	
`tenantID`	Die Mandanten-ID aus einer App-Registrierung Optional, wenn verwaltete Identitäten oder Cloud-Identität auf einem AKS-Cluster verwendet wird.	
`clientID`	Die Client-ID aus einer App-Registrierung Optional, wenn verwaltete Identitäten oder Cloud-Identität auf einem AKS-Cluster verwendet wird.	
`clientSecret`	Das Client Secret aus einer App Registration ist optional, wenn verwaltete Identitäten oder Cloud Identity auf einem AKS-Cluster verwendet werden.	
`serviceLevel`	Eines von `Standard`, `Premium` oder `Ultra`	"" (zufällig)
`location`	Name des Azure-Standorts, an dem die neuen Volumes erstellt werden Optional, wenn verwaltete Identitäten in einem AKS-Cluster aktiviert sind.	
`resourceGroups`	Liste der Ressourcengruppen zum Filtern entdeckter Ressourcen	"[]" (kein Filter)
`netappAccounts`	Liste der NetApp Konten zum Filtern der gefundenen Ressourcen	"[]" (kein Filter)
`capacityPools`	Liste der Kapazitätspools zum Filtern der gefundenen Ressourcen	"[]" (kein Filter, zufällig)

Parameter	Beschreibung	Standard
`virtualNetwork`	Name eines virtuellen Netzwerks mit einem delegierten Subnetz	""
`subnet`	Name eines delegierten Subnetzes an `Microsoft.Netapp/volumes`	""
`networkFeatures`	Satz von VNet-Funktionen für ein Volume, kann `Basic` oder `Standard` sein. Network Features ist nicht in allen Regionen verfügbar und muss möglicherweise in einem Abonnement aktiviert werden. Die Angabe von `networkFeatures` wenn die Funktionalität nicht aktiviert ist, führt dazu, dass die Volume-Bereitstellung fehlschlägt.	""
`nfsMountOptions`	Detaillierte Steuerung der NFS-Mountoptionen. Wird für SMB-Volumes ignoriert. Um Volumes mit NFS Version 4.1 zu mounten, schließen Sie `nfsvers=4` in die durch Kommas getrennte Liste der Mountoptionen ein, um NFS v4.1 auszuwählen. Mountoptionen, die in einer Speicherklassendefinition festgelegt sind, überschreiben Mountoptionen, die in der Backend-Konfiguration festgelegt sind.	"nfsvers=3"
`limitVolumeSize`	Fehler bei der Bereitstellung, wenn die angeforderte Volume-Größe über diesem Wert liegt	"" (wird nicht standardmäßig erzwungen)
`debugTraceFlags`	Debug-Flags zur Verwendung bei der Fehlersuche. Beispiel, `\{"api": false, "method": true, "discovery": true}`. Verwenden Sie dies nur, wenn Sie eine detaillierte Protokollausgabe zur Fehlersuche benötigen.	null
`nasType`	Konfigurieren Sie die Erstellung von NFS- oder SMB-Volumes. Optionen sind `nfs`, `smb` oder null. Die Einstellung auf null verwendet standardmäßig NFS-Volumes.	`nfs`
`supportedTopologies`	Stellt eine Liste der Regionen und Zonen dar, die von diesem Backend unterstützt werden. Weitere Informationen finden sich unter "CSI-Topologie verwenden".	

Parameter	Beschreibung	Standard
`qosType`	Stellt den QoS-Typ dar: Auto oder manuell.	Automatisch
`maxThroughput`	Legt den maximal zulässigen Durchsatz in MiB/sec. fest. Wird nur für manuelle QoS-Kapazitätspools unterstützt.	`4 MiB/sec`

Weitere Informationen zu Netzwerkfunktionen finden Sie unter "Netzwerkfunktionen für ein Azure NetApp Files-Volume konfigurieren".

Berücksichtigen Sie Azure-Cloud-Umgebungen (26.02)

Ab der Version 26.02 unterstützt Trident das Erstellen und Verwalten von Azure NetApp Files-Backends in mehreren Azure-Cloud-Umgebungen.

Unterstützte Azure-Clouds umfassen:

- Azure Commercial
- Azure Government (Azure Government / MAG)

Stellen Sie bei der Bereitstellung von Trident oder der Erstellung eines Azure NetApp Files-Backends sicher, dass Azure Resource Manager und Authentifizierung mit Ihrer Azure-Cloudumgebung übereinstimmen. Stimmen die Endpunkte nicht überein, `tridentctl` kann keine Authentifizierung durchgeführt werden und die Backend-Erstellung schlägt fehl.

Erforderliche Berechtigungen und Ressourcen

Wenn beim Erstellen eines PVC die Fehlermeldung „Keine Kapazitätspools gefunden“ angezeigt wird, verfügt Ihre App-Registrierung wahrscheinlich nicht über die erforderlichen Berechtigungen und Ressourcen (Subnetz, virtuelles Netzwerk, Kapazitätspool), die zugeordnet sind. Wenn Debug aktiviert ist, protokolliert Trident die beim Erstellen des Backends ermittelten Azure-Ressourcen. Stellen Sie sicher, dass eine geeignete Rolle verwendet wird.

Die Werte für `resourceGroups`, `netappAccounts`, `capacityPools`, `virtualNetwork` und `subnet` können mithilfe von Kurznamen oder vollqualifizierten Namen angegeben werden. Vollqualifizierte Namen werden in den meisten Situationen empfohlen, da Kurznamen mehreren Ressourcen mit demselben Namen zugeordnet werden können.

Wenn sich das vNet in einer anderen Ressourcengruppe als das Azure NetApp Files (ANF)-Speicherkonto befindet, geben Sie die Ressourcengruppe für das virtuelle Netzwerk bei der Konfiguration der resourceGroups-Liste für das Backend an.

Die `resourceGroups`, `netappAccounts`, und `capacityPools` Werte sind Filter, die die Menge der gefundenen Ressourcen auf diejenigen beschränken, die diesem Storage-Backend zur Verfügung stehen, und können in beliebiger Kombination angegeben werden. Vollqualifizierte Namen folgen diesem Format:

Typ	Format
Ressourcengruppe	<resource group>
NetApp-Konto	<resource group>/<netapp account>

Typ	Format
Kapazitätspool	<resource group>/<netapp account>/<capacity pool>
Virtuelles Netzwerk	<resource group>/<virtual network>
Subnetz	<resource group>/<virtual network>/<subnet>

Volumenbereitstellung

Sie können die Standard-Volume-Bereitstellung steuern, indem Sie die folgenden Optionen in einem speziellen Abschnitt der Konfigurationsdatei angeben. Siehe Beispielkonfigurationen für Details.

Parameter	Beschreibung	Standard
`exportRule`	Exportregeln für neue Volumes. `exportRule` muss eine durch Kommas getrennte Liste beliebiger Kombinationen von IPv4-Adressen oder IPv4-Subnetzen in CIDR-Notation sein. Wird für SMB-Volumes ignoriert.	"0.0.0.0/0"
`snapshotDir`	Zugriff auf das `.snapshot` Verzeichnis	`true`, `false` (explizit festlegen).
`size`	Die Standardgröße neuer Volumes	"100G"
`unixPermissions`	Die Unix-Berechtigungen neuer Volumes (4 Oktalstellen). Wird für SMB-Volumes ignoriert.	"" (Vorschaufunktion, erfordert Whitelisting im Abonnement)

Beispielkonfigurationen

Die folgenden Beispiele zeigen Basiskonfigurationen, bei denen die meisten Parameter auf Standardwerte eingestellt sind. Dies ist die einfachste Methode, ein Backend zu definieren.

Minimale Konfiguration

Dies ist die absolute Minimalkonfiguration für das Backend. Mit dieser Konfiguration erkennt Trident alle Ihre NetApp Konten, Kapazitätspools und Subnetze, die an Azure NetApp Files im konfigurierten Speicherort delegiert sind, und platziert neue Volumes zufällig in einem dieser Pools und Subnetze. Da `nasType` ausgelassen wird, gilt die `nfs` Standardeinstellung und das Backend stellt NFS-Volumes bereit.

Diese Konfiguration ist ideal, wenn Sie gerade erst mit Azure NetApp Files beginnen und Dinge ausprobieren, aber in der Praxis werden Sie eine zusätzliche Bereichsdefinition für die von Ihnen bereitgestellten Volumes wünschen.

```
---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-anf-1
  namespace: trident
spec:
  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
```

Verwaltete Identitäten für AKS

Diese Backend-Konfiguration lässt `subscriptionID`, `tenantID`, `clientID` und `clientSecret` aus, die bei der Verwendung verwalteter Identitäten optional sind.

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-anf-1
  namespace: trident
spec:
  version: 1
  storageDriverName: azure-netapp-files
  capacityPools:
    - resource-group-1/netapp-account-1/ultra-pool
  resourceGroups:
    - resource-group-1
  netappAccounts:
    - resource-group-1/netapp-account-1
  virtualNetwork: resource-group-1/eastus-prod-vnet
  subnet: resource-group-1/eastus-prod-vnet/eastus-anf-subnet
```

Cloud-Identität für AKS

Diese Backend-Konfiguration lässt `tenantID`, `clientID` und `clientSecret` aus, die bei Verwendung einer Cloud-Identität optional sind.

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-anf-1
  namespace: trident
spec:
  version: 1
  storageDriverName: azure-netapp-files
  capacityPools:
    - ultra-pool
  resourceGroups:
    - aks-ami-eastus-rg
  netappAccounts:
    - smb-na
  virtualNetwork: eastus-prod-vnet
  subnet: eastus-anf-subnet
  location: eastus
  subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
```

Spezifische Service-Level-Konfiguration mit Kapazitätspoolfiltern

Diese Backend-Konfiguration platziert Volumes im `eastus` Standort von Azure in einem `Ultra` Kapazitätspool. Trident erkennt automatisch alle an Azure NetApp Files in diesem Standort delegierten Subnetze und platziert ein neues Volume zufällig auf einem davon.

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

Backend-Beispiel mit manuellen QoS-Kapazitätspools

Diese Backend-Konfiguration platziert Volumes im `eastus` Standort von Azure mit manuellen QoS-Kapazitätspools.

```
---
version: 1
storageDriverName: azure-netapp-files
backendName: anf1
location: eastus
labels:
  clusterName: test-cluster-1
  cloud: anf
  nasType: nfs
defaults:
  qosType: Manual
storage:
  - serviceLevel: Ultra
    labels:
      performance: gold
    defaults:
      maxThroughput: 10
  - serviceLevel: Premium
    labels:
      performance: silver
    defaults:
      maxThroughput: 5
  - serviceLevel: Standard
    labels:
      performance: bronze
    defaults:
      maxThroughput: 3
```

Erweiterte Konfiguration

Diese Backend-Konfiguration beschränkt den Umfang der Volume-Platzierung weiter auf ein einzelnes Subnetz und ändert außerdem 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: application-group-1/eastus-prod-vnet
subnet: application-group-1/eastus-prod-vnet/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"
```

Konfiguration des virtuellen Pools

Diese Backend-Konfiguration definiert mehrere Speicherpools in einer einzigen Datei. Dies ist nützlich, wenn Sie mehrere Kapazitätspools haben, die unterschiedliche Service-Level unterstützen und Sie Speicherklassen in Kubernetes erstellen möchten, die diese repräsentieren. Virtuelle Pool-Labels wurden verwendet, um die Pools anhand von `performance`zu unterscheiden.

```
---
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:
      - application-group-1/netapp-account-1/ultra-1
      - application-group-1/netapp-account-1/ultra-2
    networkFeatures: Standard
  - labels:
      performance: silver
    serviceLevel: Premium
    capacityPools:
      - application-group-1/netapp-account-1/premium-1
  - labels:
      performance: bronze
    serviceLevel: Standard
    capacityPools:
      - application-group-1/netapp-account-1/standard-1
      - application-group-1/netapp-account-1/standard-2
```

Unterstützte Topologiekonfiguration

Trident erleichtert die Bereitstellung von Volumes für Workloads basierend auf Regionen und Verfügbarkeitszonen. Der `supportedTopologies` Block in dieser Backend-Konfiguration wird verwendet, um eine Liste von Regionen und Zonen pro Backend bereitzustellen. Die hier angegebenen Regions- und Zonenwerte müssen mit den Regions- und Zonenwerten aus den Labels auf jedem Kubernetes-Clusterknoten übereinstimmen. Diese Regionen und Zonen stellen die Liste der zulässigen Werte dar, die in einer Speicherklasse angegeben werden können. Für Speicherklassen, die eine Teilmenge der in einem Backend bereitgestellten Regionen und Zonen enthalten, erstellt Trident Volumes in der genannten Region und Zone. Weitere Informationen finden sich unter "CSI-Topologie verwenden".

```
---
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
supportedTopologies:
  - topology.kubernetes.io/region: eastus
    topology.kubernetes.io/zone: eastus-1
  - topology.kubernetes.io/region: eastus
    topology.kubernetes.io/zone: eastus-2
```

Speicherklassendefinitionen

Die folgenden `StorageClass` Definitionen beziehen sich auf die oben genannten Speicherpools.

Beispieldefinitionen unter Verwendung des `parameter.selector` Felds

Mit `parameter.selector` können Sie für jedes `StorageClass` den virtuellen Pool angeben, der zum Hosten eines Volumes verwendet wird. Das Volume wird die im gewählten Pool definierten Aspekte aufweisen.

```
---
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

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

Grundkonfiguration 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"
```

Verwendung unterschiedlicher Geheimnisse pro Namensraum

```
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}
```

Verwendung unterschiedlicher Secrets pro Volume

```
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`filtert für NFS-Pools.

Das Backend erstellen

Nachdem Sie die Backend-Konfigurationsdatei erstellt haben, führen Sie den folgenden Befehl aus:

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

Wenn Sie eine nicht-kommerzielle Azure-Cloud verwenden, stellen Sie sicher, dass `tridentctl` für die Verwendung des Azure Resource Manager und der Authentifizierung-Endpunkte für Ihre Azure-Cloudumgebung konfiguriert ist. Wenn die Backend-Erstellung fehlschlägt, überprüfen Sie Ihre Backend-Konfiguration und sehen Sie sich die Protokolle an, um die Ursache zu ermitteln:

```
tridentctl logs
```

Nachdem Sie das Problem mit der Konfigurationsdatei identifiziert und behoben haben, können Sie den create-Befehl erneut ausführen.

Google Cloud NetApp Volumes

Google Cloud NetApp Volumes konfigurieren

Sie können Google Cloud NetApp Volumes als Backend für Trident konfigurieren, um Speicher für Kubernetes-Workloads bereitzustellen.

Überblick

Trident unterstützt Google Cloud NetApp Volumes sowohl für NAS- (NFS und SMB) als auch für Block-Workloads (iSCSI).

- NAS-Workloads nutzen das `google-cloud-netapp-volumes` Backend
- Block-Workloads (iSCSI) nutzen das `google-cloud-netapp-volumes-san` Backend

NAS-Volumes bieten dateibasierten Speicher und werden über NFS- oder SMB-Protokolle angesprochen. Diese Volumes unterstützen den gemeinsamen Zugriff durch mehrere Pods oder Knoten.

Block-Volumes bieten rohen Blockspeicher und werden als iSCSI-Geräte angesprochen, die an Kubernetes-Knoten angeschlossen sind. Diese Volumes werden verwendet, wenn Anwendungen Zugriff auf Blockebene benötigen.

Dies gilt für die folgenden Umgebungen:

- Trident 26.02 und später
- Google Kubernetes Engine (GKE) oder Red Hat OpenShift
- Google Cloud NetApp Volumes-Speicherpools

Informationen zur Konfiguration von Block-Speicher (iSCSI) finden Sie unter "Blockspeicher konfigurieren

(iSCSI)".

Bereiten Sie die Konfiguration vor

Cloud Identity ermöglicht es Kubernetes-Workloads, auf Google Cloud-Ressourcen zuzugreifen, indem sie sich als Workload-Identität authentifizieren, anstatt statische Anmeldeinformationen zu verwenden.

Um Cloud Identity mit Google Cloud NetApp Volumes zu verwenden, müssen Sie Folgendes haben:

- Ein mit Google Kubernetes Engine (GKE) bereitgestellter Kubernetes-Cluster
- Workload-Identität im GKE-Cluster aktiviert und Metadatenserver auf den Knotenpools aktiviert
- Ein Google Cloud-Dienstkonto mit der Google Cloud NetApp Volumes Admin-Rolle (`roles/netapp.admin` oder einer entsprechenden benutzerdefinierten Rolle
- Trident wurde mit dem Cloud-Anbieter auf `GCP` gesetzt und der Cloud-Identitätsannotation konfiguriert installiert

Trident Operator

Um Trident mithilfe des Trident Operators zu installieren, bearbeiten Sie `tridentorchestrator_cr.yaml`:

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  namespace: trident
  cloudProvider: "GCP"
  cloudIdentity: "iam.gke.io/gcp-service-account: cloudvolumes-admin-
sa@mygcpproject.iam.gserviceaccount.com"
```

Helm

Legen Sie beim Installieren von Trident mit Helm den Cloud-Anbieter und die Cloud-Identität fest:

```
helm install trident trident-operator-100.6.0.tgz \
  --set cloudProvider=GCP \
  --set cloudIdentity="iam.gke.io/gcp-service-account: cloudvolumes-
admin-sa@mygcpproject.iam.gserviceaccount.com"
```

tridentctl

Installieren Sie Trident, indem Sie den Cloud-Anbieter und die Cloud-Identität angeben:

```
tridentctl install \
  --cloud-provider=GCP \
  --cloud-identity="iam.gke.io/gcp-service-account: cloudvolumes-admin-
sa@mygcpproject.iam.gserviceaccount.com" \
  -n trident
```

NAS-Speicher konfigurieren

> Für Google Cloud NetApp Volumes UNIFIED-Speicherpools wendet Trident während der Volume-Operationen UNIFIED-spezifische Namens- und Validierungsregeln an.

Beim Auffinden eines Volumes kann Trident mehrere kompatible Varianten des Volumennamens (z. B. Bindestrich- und Unterstrichformate) auswerten, um die Zuverlässigkeit von Import und Erkennung zu verbessern.

Treiberdetails

Trident stellt den `google-cloud-netapp-volumes` Treiber zur Verfügung, um NAS-Speicher von Google

Cloud NetApp Volumes bereitzustellen.

Der Treiber unterstützt die folgenden Zugriffsmodi:

- ReadWriteOnce (RWO)
- ReadOnlyMany (ROX)
- ReadWriteMany (RWX)
- ReadWriteOncePod (RWOP)

Treiber	Protokoll	volumeMode	Unterstützte Zugriffsmodi	Unterstützte Dateisysteme
`google-cloud-netapp-volumes`	NFS SMB	Dateisystem	RWO, ROX, RWX, RWOP	`nfs`, `smb`

Konfigurieren Sie ein Trident NAS-Backend

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: gcnv-nas
  namespace: trident
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: "<project-number>"
  location: "<region>"
  sdkTimeout: "600"
  storage:
  - labels:
      cloud: gcp
    network: "<vpc-network>"
```

NAS-Volumen bereitstellen

NAS-Volumes werden mit dem `google-cloud-netapp-volumes` Backend bereitgestellt und unterstützen die Protokolle NFS und SMB.

StorageClass für NFS-Volumes

Um NFS-Volumes bereitzustellen, setzen Sie `nasType` auf `nfs`.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gcnv-nfs
provisioner: csi.trident.netapp.io
parameters:
  backendType: "google-cloud-netapp-volumes"
  trident.netapp.io/nasType: "nfs"
allowVolumeExpansion: true
```

StorageClass für SMB-Volumes

Um SMB-Volumes bereitzustellen, setzen Sie `nasType` auf `smb` und geben Sie die Anmeldeinformationen an.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gcnv-smb
provisioner: csi.trident.netapp.io
parameters:
  backendType: "google-cloud-netapp-volumes"
  trident.netapp.io/nasType: "smb"
  csi.storage.k8s.io/node-stage-secret-name: "smbcreds"
  csi.storage.k8s.io/node-stage-secret-namespace: "default"
allowVolumeExpansion: true
```

PersistentVolumeClaim-Beispiel (RWX)

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: gcnv-nas-rwx
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 100Gi
  storageClassName: gcnv-nfs
```

PersistentVolumeClaim-Beispiel (RWO)

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: gcnv-nas-rwo
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 100Gi
  storageClassName: gcnv-nfs
```

NAS-Volumes verwenden `volumeMode: Filesystem`.

Google Cloud NetApp Volumes für SAN-Workloads konfigurieren

Sie können Trident so konfigurieren, dass Block-Storage-Volumes über das iSCSI-Protokoll von Google Cloud NetApp Volumes bereitgestellt werden. SAN-Volumes werden mithilfe des `google-cloud-netapp-volumes-san` Speichertreibers aus Flex Unified Storage Pools bereitgestellt.

Dieser Treiber ist für Block-Workloads ausgelegt und unterstützt keine NAS-Protokolle.

Das `google-cloud-netapp-volumes-san` Backend ist erforderlich, um iSCSI-Blockvolumes bereitzustellen. Das `google-cloud-netapp-volumes` Backend unterstützt nur NAS-Protokolle und kann nicht für SAN-Workloads verwendet werden.

Überblick

Trident unterstützt Google Cloud NetApp Volumes SAN (iSCSI)-Workloads mithilfe des `google-cloud-netapp-volumes-san` Treibers.

SAN-Volumes werden aus Flex Unified-Speicherpools bereitgestellt und den Kubernetes-Knoten als iSCSI-Blockgeräte präsentiert.

Dies gilt für die folgenden Umgebungen:

- Trident 26.02 und später
- Google Kubernetes Engine (GKE) oder Red Hat OpenShift
- Google Cloud NetApp Volumes Flex – Einheitliche Speicherpools
- iSCSI-basierte Workloads

Flex Unified Speicherpools

Flex Unified-Speicherpools bieten Blockspeicher über das iSCSI-Protokoll und sind für die SAN-Bereitstellung erforderlich:

- Flex Unified REGIONAL-Pools werden unterstützt.
- Flex Unified ZONAL-Pools werden ab Trident 26.02.1 unterstützt.
- Für SAN-Workloads wird ausschließlich die Serviceebene **Flex** unterstützt.

Konfigurieren Sie ein Trident SAN-Backend

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: gcnv-san
  namespace: trident
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes-san
  projectNumber: "<project-number>"
  location: "<region>"
  sdkTimeout: "600"
  storage:
  - labels:
      cloud: gcp
      performance: flex
    network: "<vpc-network>"
    serviceLevel: Flex
```

Erstellen Sie eine StorageClass

Nach der Konfiguration des SAN-Backends erstellen Sie eine StorageClass, die auf den `google-cloud-netapp-volumes-san` Treiber verweist.

Der Dateisystemtyp wird in der StorageClass definiert, nicht im Backend.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gcnv-san
provisioner: csi.trident.netapp.io
parameters:
  backendType: "google-cloud-netapp-volumes-san"
  fsType: "ext4"
allowVolumeExpansion: true
```

Unterstützte Dateisystemtypen:

- `ext4` (Standard)
- `ext3`

- `xfs`

> Der SAN-Treiber unterstützt ausschließlich das Flex-Servicelevel und verwendet keine NAS-spezifischen Backend-Parameter wie `exportRule`, `unixPermissions`, `nasType`, `snapshotDir`, `nfsMountOptions` oder tiering-bezogene Einstellungen.

Block-Volumes bereitstellen

ReadWriteOnce (RWO)

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: gcnv-san-rwo
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 100Gi
  storageClassName: gcnv-san
```

ReadWriteOncePod (RWOP)

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: gcnv-san-rwop
spec:
  accessModes:
    - ReadWriteOncePod
  resources:
    requests:
      storage: 100Gi
  storageClassName: gcnv-san
```

ReadOnlyMany (ROX)

Ein gängiges Muster für ROX ist es, ein vorhandenes ReadWriteOnce-Volume zu klonen und den Klon als schreibgeschützt einzubinden.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: gcnv-san-rox
spec:
  accessModes:
    - ReadOnlyMany
  resources:
    requests:
      storage: 100Gi
  storageClassName: gcnv-san
  dataSource:
    kind: PersistentVolumeClaim
    name: gcnv-san-rwo
```

ReadWriteMany (RWX) — nur Rohdatenblock

ReadWriteMany wird nur unterstützt, wenn `volumeMode: Block`.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: gcnv-san-raw-rwx
spec:
  accessModes:
    - ReadWriteMany
  volumeMode: Block
  resources:
    requests:
      storage: 100Gi
  storageClassName: gcnv-san
```

Blockvolumenverhalten

Block-Volumes werden als iSCSI-LUNs bereitgestellt und den Kubernetes-Knoten als Blockgeräte präsentiert.

Block-Volumes:

- Verwenden Sie das iSCSI-Protokoll
- Unterstützung der Dateisystem- und Rohblockdarstellung
- Werden von Trident angehängt und verwaltet
- Mehrere Kubernetes-Zugriffsmodi unterstützen

Zugriffsmodi

Von Trident bereitgestellte Block-Volumes unterstützen die folgenden Zugriffsmodi:

- `ReadWriteOnce` (RWO)
- `ReadOnlyMany` (ROX)
- `ReadWriteOncePod` (RWOP)
- `ReadWriteMany` (RWX), wird nur unterstützt, wenn `volumeMode: Block`

volumeMode-Verhalten

Das `volumeMode` Feld steuert, wie ein Blockvolumen angezeigt wird:

- `Filesystem` Trident formatiert und mountet das Volume.
- `Block` Trident verbindet das Gerät und stellt es als Raw-Blockgerät bereit.

Unterstützte Operationen

Blockvolumes, die mithilfe des `google-cloud-netapp-volumes-san` Treibers bereitgestellt werden:

- Erstellen
- Löschen
- Klonen
- Schnappschuss
- Größe ändern
- Importieren

Zusätzliches GiB-Überprovisionierungsverhalten

Google Cloud NetApp Volumes Blockvolumes beinhalten einen internen Metadaten-Overhead. Dieser Overhead reduziert die vom Kernel sichtbare Gerätegröße im Vergleich zur bereitgestellten Kapazität.

Tests zeigen:

- Ungefähr 300 KiB Overhead bei der erstmaligen Erstellung
- Bis zu ca. 107 MiB Overhead nach einer Größenänderung

Da Google Cloud NetApp Volumes nur ganze GiB-Zuweisungen akzeptiert, stellt Trident sicher, dass die nutzbare Gerätegröße die PVC-Anforderung immer erfüllt oder übertrifft, indem:

- Aufrunden der angeforderten Größe auf das nächste ganze GiB
- Hinzufügen eines zusätzlichen 1 GiB Puffers

Beispiel:

- PVC-Anfrage: 100 GiB
- Bereitgestellte Größe in Google Cloud NetApp Volumes: 101 GiB
- Für die Anwendung sichtbarer nutzbarer Speicherplatz: mindestens 100 GiB

Pod-Beispiele

Dateisystemgebundenes Blockvolume (RWO)

```
apiVersion: v1
kind: Pod
metadata:
  name: app-rwo
spec:
  containers:
  - name: app
    image: ubuntu:22.04
    command: ["sleep", "infinity"]
    volumeMounts:
    - name: data
      mountPath: /mnt/data
  volumes:
  - name: data
    persistentVolumeClaim:
      claimName: gcnv-san-rwo
```

Raw-Block-Gerät (RWX)

```
apiVersion: v1
kind: Pod
metadata:
  name: app-raw-rwx
spec:
  containers:
  - name: app
    image: ubuntu:22.04
    command: ["sleep", "infinity"]
    volumeDevices:
    - name: data
      devicePath: /dev/xda
  volumes:
  - name: data
    persistentVolumeClaim:
      claimName: gcnv-san-raw-rwx
```

Anfüge- und Einhängeverhalten

Für SAN-Volumes, die von Google Cloud NetApp Volumes bereitgestellt wurden:

- Trident erstellt eine Logical Unit Number (LUN) in einem Flex Unified Speicherpool.

- Während der Veröffentlichung ordnet Trident die LUN einer Hostgruppe pro Knoten zu.
- Während der Knoten-Staging-Phase führt Trident Folgendes aus:
  - Meldet sich am iSCSI-Ziel an
  - Entdeckt die LUN
  - Konfiguriert Multipath
- Wenn `volumeMode: Filesystem` erforderlich, formatiert Trident das Gerät und bindet es ein.
- Wenn `volumeMode: Block`Trident das Gerät anschließt, wird es ohne Formatierung oder Einbindung direkt dem Pod bereitgestellt.

> SAN-Blockvolumes bieten keine verteilte Sperrung oder Schreibkoordination. Wenn auf ein Blockvolume von mehreren Knoten (ReadWriteMany mit `volumeMode: Block`) zugegriffen wird, muss die Anwendung oder das Dateisystem die Parallelität verwalten.

Bereiten Sie die Konfiguration eines Google Cloud NetApp Volumes-Backends vor

Bevor Sie Ihr Google Cloud NetApp Volumes-Backend konfigurieren können, müssen Sie sicherstellen, dass die folgenden Voraussetzungen erfüllt sind.

Voraussetzungen für NFS- oder SMB-Volumes

Wenn Sie Google Cloud NetApp Volumes zum ersten Mal oder an einem neuen Standort verwenden, ist eine anfängliche Konfiguration erforderlich, um Google Cloud NetApp Volumes einzurichten und ein NFS- oder SMB-Volume zu erstellen. Siehe "Bevor Sie beginnen".

Stellen Sie sicher, dass Sie Folgendes haben, bevor Sie das Google Cloud NetApp Volumes-Backend konfigurieren:

- Ein Google Cloud-Konto, das mit dem Google Cloud NetApp Volumes Service konfiguriert ist. Siehe "Google Cloud NetApp Volumes".
- Projektnummer Ihres Google Cloud account. Siehe "Projekte identifizieren".
- Ein Google Cloud-Dienstkonto mit der NetApp Volumes Admin (`roles/netapp.admin`-Rolle. Siehe "Identitäts- und Zugriffsmanagement-Rollen und -Berechtigungen".
- API-Schlüsseldatei für Ihr GCNV-Konto. Siehe "Erstellen Sie einen Dienstkontoschlüssel"
- Ein Speicherpool. Siehe "Übersicht der Storage Pools".

Weitere Informationen zum Einrichten des Zugriffs auf Google Cloud NetApp Volumes finden Sie unter "Richten Sie den Zugriff auf Google Cloud NetApp Volumes ein".

Google Cloud NetApp Volumes Backend-Konfigurationsoptionen und Beispiele

Erfahren Sie mehr über Backend-Konfigurationsoptionen für Google Cloud NetApp Volumes und sehen Sie sich Konfigurationsbeispiele an.

Backend-Konfigurationsoptionen

Jedes Backend stellt Volumes in einer einzelnen Google Cloud Region bereit. Um Volumes in anderen Regionen zu erstellen, können Sie zusätzliche Backends definieren.

Parameter	Beschreibung	Standard
`version`		Immer 1
`storageDriverName`	Name des Speichertreibers	Der Wert `storageDriverName` muss als "google-cloud-netapp-volumes" angegeben werden.
`backendName`	(Optional) Benutzerdefinierter Name des Storage-Backends	Driver name + "_" + Teil des API-Schlüssels
`storagePools`	Optionaler Parameter, der verwendet wird, um Speicherpools für die Volume-Erstellung anzugeben.	
`projectNumber`	Google Cloud-Konto-Projektnummer. Der Wert ist auf der Startseite des Google Cloud-Portals zu finden.	
`location`	Der Google Cloud-Standort, an dem Trident GCNV-Volumes erstellt. Beim Erstellen regionsübergreifender Kubernetes-Cluster können Volumes, die in einem `location` erstellt wurden, in Workloads verwendet werden, die auf Knoten in mehreren Google Cloud-Regionen geplant sind. Für regionsübergreifenden Datenverkehr fallen zusätzliche Kosten an.	
`apiKey`	API-Schlüssel für das Google Cloud-Dienstkonto mit der `netapp.admin` Rolle. Er enthält den JSON-formatierten Inhalt der privaten Schlüsseldatei eines Google Cloud-Dienstkontos (wortwörtlich in die Backend-Konfigurationsdatei kopiert). Die `apiKey` muss Schlüssel-Wert-Paare für die folgenden Schlüssel enthalten: `type`, `project_id`, `client_email`, `client_id`, `auth_uri`, `token_uri`, `auth_provider_x509_cert_url` und `client_x509_cert_url`.	
`nfsMountOptions`	Detaillierte Steuerung der NFS-Mountoptionen.	"nfsvers=3"
`limitVolumeSize`	Die Bereitstellung schlägt fehl, wenn die angeforderte Volume-Größe über diesem Wert liegt.	"" (wird nicht standardmäßig erzwungen)
`serviceLevel`	Der Servicegrad eines Speicherpools und seiner Volumes. Die Werte sind `flex`, `standard`, `premium` oder `extreme`.	
`labels`	Satz beliebiger JSON-formatierter Bezeichnungen, die auf Volumes angewendet werden sollen	""
`network`	Google Cloud-Netzwerk, das für GCNV-Volumes verwendet wird.	
`debugTraceFlags`	Debug-Flags zur Verwendung bei der Fehlersuche. Beispiel, `{"api":false, "method":true}`. Verwenden Sie dies nur, wenn Sie eine detaillierte Protokollausgabe zur Fehlersuche benötigen.	null

Parameter	Beschreibung	Standard
`nasType`	Konfigurieren Sie die Erstellung von NFS- oder SMB-Volumes. Optionen sind `nfs`, `smb` oder null. Die Einstellung auf null verwendet standardmäßig NFS-Volumes.	`nfs`
`supportedTopologies`	Stellt eine Liste der Regionen und Zonen dar, die von diesem Backend unterstützt werden. Weitere Informationen finden sich unter "CSI-Topologie verwenden". Zum Beispiel: `supportedTopologies: - topology.kubernetes.io/region: asia-east1 topology.kubernetes.io/zone: asia-east1-a`	

Optionen zur Volumenbereitstellung

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

Parameter	Beschreibung	Standard
`exportRule`	Die Exportregeln für neue Volumes. Muss eine durch Kommas getrennte Liste beliebiger IPv4-Adressen sein.	"0.0.0.0/0"
`snapshotDir`	Zugriff auf das `.snapshot` Verzeichnis	`true`, `false` (Das Standardverhalten kann abweichen. Stellen Sie es explizit auf „false“ für NFSv3 ein
`snapshotReserve`	Prozentsatz des für Snapshots reservierten Volumens	"" (Standardwert von 0 akzeptieren)
`unixPermissions`	Die Unix-Berechtigungen neuer Volumes (4 Oktalstellen).	""

Beispielkonfigurationen

Die folgenden Beispiele zeigen Basiskonfigurationen, bei denen die meisten Parameter auf Standardwerte eingestellt sind. Dies ist die einfachste Methode, ein Backend zu definieren.

Minimale Konfiguration

Dies ist die absolute Minimalkonfiguration für das Backend. Mit dieser Konfiguration erkennt Trident alle Ihre an Google Cloud NetApp Volumes delegierten Speicherpools am konfigurierten Standort und platziert neue Volumes zufällig in einem dieser Pools. Da `nasType` ausgelassen wird, gilt die `nfs` Standardeinstellung und das Backend stellt NFS-Volumes bereit.

Diese Konfiguration ist ideal, wenn Sie gerade erst mit Google Cloud NetApp Volumes beginnen und Dinge ausprobieren, aber in der Praxis müssen Sie möglicherweise zusätzliche Einschränkungen für die von Ihnen bereitgestellten Volumes festlegen.

Ersetzen Sie `<id_value>` und `<key_value>` durch Ihre Servicekonto-Zugangsdaten.

```
---
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-gcnv-secret
type: Opaque
stringData:
  private_key_id: "<id_value>"
  private_key: |
    -----BEGIN PRIVATE KEY-----
    <key_value>
    -----END PRIVATE KEY-----
---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-gcnv
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: "123455380079"
  location: europe-west6
  serviceLevel: premium
  apiKey:
    type: service_account
    project_id: my-gcnv-project
    client_email: myproject-prod@my-gcnv-
project.iam.gserviceaccount.com
    client_id: "103346282737811234567"
    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/myproject-prod%40my-
gcnv-project.iam.gserviceaccount.com
  credentials:
    name: backend-tbc-gcnv-secret
```

Konfiguration für SMB-Volumes

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-gcnv1
  namespace: trident
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: "123456789"
  location: asia-east1
  serviceLevel: flex
  nasType: smb
  apiKey:
    type: service_account
    project_id: cloud-native-data
    client_email: trident-sample@cloud-native-
data.iam.gserviceaccount.com
    client_id: "123456789737813416734"
    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/trident-
sample%40cloud-native-data.iam.gserviceaccount.com
  credentials:
    name: backend-tbc-gcnv-secret
```

Konfiguration mit StoragePools Filter

```
---
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-gcnv-secret
type: Opaque
stringData:
  private_key_id: "<id_value>"
  private_key: |
    -----BEGIN PRIVATE KEY-----
    <key_value>
    -----END PRIVATE KEY-----
---

---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-gcnv
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: "123455380079"
  location: europe-west6
  serviceLevel: premium
  storagePools:
    - premium-pool1-europe-west6
    - premium-pool2-europe-west6
  apiKey:
    type: service_account
    project_id: my-gcnv-project
    client_email: myproject-prod@my-gcnv-
project.iam.gserviceaccount.com
    client_id: "103346282737811234567"
    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/myproject-prod%40my-
gcnv-project.iam.gserviceaccount.com
  credentials:
    name: backend-tbc-gcnv-secret
```

Konfiguration des virtuellen Pools

Diese Backend-Konfiguration definiert mehrere virtuelle Pools in einer einzigen Datei. Virtuelle Pools werden im `storage` Abschnitt definiert. Sie sind nützlich, wenn Sie mehrere Speicherpools mit unterschiedlichen Service-Levels haben und entsprechende Speicherklassen in Kubernetes erstellen möchten, die diese repräsentieren. Virtuelle Pool-Labels dienen zur Unterscheidung der Pools. Im folgenden Beispiel werden beispielsweise das `performance` Label und der `serviceLevel` Typ verwendet, um virtuelle Pools zu unterscheiden.

Sie können auch Standardwerte festlegen, die für alle virtuellen Pools gelten, und die Standardwerte einzelner virtueller Pools überschreiben. Im folgenden Beispiel dienen `snapshotReserve` und `exportRule` als Standardwerte für alle virtuellen Pools.

Weitere Informationen finden sich unter "Virtuelle Pools".

```
---
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-gcnv-secret
type: Opaque
stringData:
  private_key_id: "<id_value>"
  private_key: |
    -----BEGIN PRIVATE KEY-----
    <key_value>
    -----END PRIVATE KEY-----

---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-gcnv
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: "123455380079"
  location: europe-west6
  apiKey:
    type: service_account
    project_id: my-gcnv-project
    client_email: myproject-prod@my-gcnv-
project.iam.gserviceaccount.com
    client_id: "103346282737811234567"
    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/myproject-prod%40my-
gcnv-project.iam.gserviceaccount.com
  credentials:
    name: backend-tbc-gcnv-secret
  defaults:
    snapshotReserve: "10"
    exportRule: 10.0.0.0/24
  storage:
    - labels:
        performance: extreme
      serviceLevel: extreme
      defaults:
        snapshotReserve: "5"
        exportRule: 0.0.0.0/0
    - labels:
        performance: premium
      serviceLevel: premium
    - labels:
        performance: standard
      serviceLevel: standard
```

Cloud-Identität für GKE

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-gcp-gcnv
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: '012345678901'
  network: gcnv-network
  location: us-west2
  serviceLevel: Premium
  storagePool: pool-premium1
```

Unterstützte Topologiekonfiguration

Trident erleichtert die Bereitstellung von Volumes für Workloads basierend auf Regionen und Verfügbarkeitszonen. Der `supportedTopologies` Block in dieser Backend-Konfiguration wird verwendet, um eine Liste von Regionen und Zonen pro Backend bereitzustellen. Die hier angegebenen Regions- und Zonenwerte müssen mit den Regions- und Zonenwerten aus den Labels auf jedem Kubernetes-Clusterknoten übereinstimmen. Diese Regionen und Zonen stellen die Liste der zulässigen Werte dar, die in einer Speicherklasse angegeben werden können. Für Speicherklassen, die eine Teilmenge der in einem Backend bereitgestellten Regionen und Zonen enthalten, erstellt Trident Volumes in der genannten Region und Zone. Weitere Informationen finden sich unter "CSI-Topologie verwenden".

```
---
version: 1
storageDriverName: google-cloud-netapp-volumes
subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
tenantID: 68e4f836-edc1-fake-bff9-b2d865ee56cf
clientID: dd043f63-bf8e-fake-8076-8de91e5713aa
clientSecret: SECRET
location: asia-east1
serviceLevel: flex
supportedTopologies:
  - topology.kubernetes.io/region: asia-east1
    topology.kubernetes.io/zone: asia-east1-a
  - topology.kubernetes.io/region: asia-east1
    topology.kubernetes.io/zone: asia-east1-b
```

Was kommt als Nächstes?

Nachdem Sie die Backend-Konfigurationsdatei erstellt haben, führen Sie den folgenden Befehl aus:

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

Um zu überprüfen, ob das Backend erfolgreich erstellt wurde, führen Sie den folgenden Befehl aus:

```
kubectl get tridentbackendconfig

NAME                BACKEND NAME        BACKEND UUID
PHASE   STATUS
backend-tbc-gcnv    backend-tbc-gcnv    b2fd1ff9-b234-477e-88fd-713913294f65
Bound   Success
```

Schlägt die Backend-Erstellung fehl, liegt ein Fehler in der Backend-Konfiguration vor. Sie können das Backend mit dem `kubectl get tridentbackendconfig <backend-name>`-Befehl beschreiben oder die Protokolle einsehen, um die Ursache zu ermitteln, indem Sie den folgenden Befehl ausführen:

```
tridentctl logs
```

Nachdem Sie das Problem mit der Konfigurationsdatei identifiziert und behoben haben, können Sie das Backend löschen und den Befehl zum Erstellen erneut ausführen.

Speicherklassendefinitionen

Nachfolgend ist eine grundlegende `StorageClass` Definition, die sich auf das oben genannte Backend bezieht.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gcnv-nfs-sc
provisioner: csi.trident.netapp.io
parameters:
  backendType: "google-cloud-netapp-volumes"
```

Beispieldefinitionen unter Verwendung des `parameter.selector` Feldes:

Mit `parameter.selector` können Sie für jedes `StorageClass` den "virtueller Pool" angeben, der zum Hosten eines Volumes verwendet wird. Das Volume wird die im gewählten Pool definierten Aspekte aufweisen.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: extreme-sc
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=extreme
  backendType: google-cloud-netapp-volumes

---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: premium-sc
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=premium
  backendType: google-cloud-netapp-volumes

---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: standard-sc
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=standard
  backendType: google-cloud-netapp-volumes
```

Weitere Details zu Speicherklassen finden Sie unter "Erstellen Sie eine Speicherklasse".

Beispieldefinitionen für SMB-Volumes

Mit `nasType`, `node-stage-secret-name` und `node-stage-secret-namespace` können Sie ein SMB-Volume angeben und die erforderlichen Active Directory-Anmeldeinformationen bereitstellen. Jeder Active Directory-Benutzername/jedes Passwort mit beliebigen oder keinen Berechtigungen kann für das Knotenstufengeheimnis verwendet werden.

Grundkonfiguration im Standard-Namespace

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

Verwendung unterschiedlicher Geheimnisse pro Namensraum

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

Verwendung unterschiedlicher Secrets pro Volume

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gcnv-sc-smb
provisioner: csi.trident.netapp.io
parameters:
  backendType: "google-cloud-netapp-volumes"
  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`filtert für NFS-Pools.

PVC-Definitionsbeispiel

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: gcnv-nfs-pvc
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 100Gi
  storageClassName: gcnv-nfs-sc
```

Um zu überprüfen, ob die PVC gebunden ist, führen Sie den folgenden Befehl aus:

```
kubectl get pvc gcnv-nfs-pvc

NAME          STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS AGE
gcnv-nfs-pvc  Bound    pvc-b00f2414-e229-40e6-9b16-ee03eb79a213  100Gi
RWX            gcnv-nfs-sc  1m
```

Automatisches Tiering für Google Cloud NetApp Volumes konfigurieren

Auto-Tiering wird über Trident-Backend-Parameter und PersistentVolumeClaim-Annotationen während der Volume-Bereitstellung konfiguriert. Sie können Auto-Tiering für Google Cloud NetApp Volumes mit Trident konfigurieren.

Überblick

Auto-Tiering ermöglicht es Trident, Volumes bereitzustellen, die inaktive Daten automatisch von einer Performance-Ebene auf eine Kapazitätsebene verschieben. Dadurch werden die Speicherkosten gesenkt, während die Leistung für häufig abgerufene Daten erhalten bleibt.

Trident wendet die Einstellungen für das automatische Tiering nur bei der Volume-Erstellung an. Änderungen nach der Bereitstellung werden in Trident 26.02 nicht unterstützt.

Konzepte

Automatisches Tiering

Auto-Tiering verschiebt selten genutzte Daten basierend auf den Zugriffsmustern von einer Performance-Tier auf eine Kapazitäts-Tier. Die Datenverschiebung erfolgt asynchron und ist nicht sofort.

Tiering-Richtlinie

Die Tiering-Richtlinie legt fest, ob Auto-Tiering für ein Volume aktiviert ist.

Folgende Richtlinien werden unterstützt: * `auto`: Aktiviert die automatische Stufenaufteilung basierend auf Zugriffsmustern * `none`: Deaktiviert die automatische Stufenaufteilung

Kühltage

Die Kühltage legen die Mindestanzahl an Tagen fest, die ein Datenblock inaktiv bleiben muss, bevor er für das Tiering in Frage kommt. Kühltage gelten nur, wenn die Tiering-Richtlinie auf `auto` eingestellt ist.

Konfigurationsmodell

Konfigurationsbereiche

Auto-Tiering kann auf mehreren Ebenen konfiguriert werden:

- **Geltungsbereich des Speicherpools** Gilt für alle Volumes, die aus dem Pool bereitgestellt wurden.
- **Volumenumfang** Gilt für ein einzelnes Volumen durch PersistentVolumeClaim-Annotationen.

Trident ermittelt die effektive Konfiguration basierend darauf, wo jede Einstellung definiert ist.

Konfigurationspriorität

Wenn dieselbe Einstellung in mehreren Bereichen definiert ist, wendet Trident die folgende Rangfolge an:

1. PersistentVolumeClaim Anmerkungen
2. Trident Backend-Konfiguration
3. Standardeinstellungen für Speicherpools

Einstellungen, die auf einer höheren Ebene definiert sind, überschreiben Werte auf niedrigerer Ebene.

Unterstützte Funktionalität in Trident 26.02

Trident 26.02 unterstützt die folgenden Auto-Tiering-Funktionen für Google Cloud NetApp Volumes:

- Aktivieren oder Deaktivieren des automatischen Tierings während der Volume-Bereitstellung
- Definieren einer Tiering-Richtlinie in der Trident Backend-Konfiguration
- Überschreiben der Tiering-Policy und der Cooling Days pro Volume mithilfe von PVC-Anmerkungen
- Konfiguration von Kühltagen für Volumes mit aktiviertem Auto-Tiering

Nicht unterstützte Funktionalität in Trident 26.02

Die folgenden Operationen werden nicht unterstützt:

- Ändern der Auto-Tiering-Einstellungen nach Volume-Erstellung
- Ändern von Tiering-Richtlinien auf bestehenden Volumes mithilfe von Kubernetes-Updates
- Anwenden von Auto-Tiering-Einstellungen außerhalb von Trident-verwalteten Bereitstellungsworkflows

Backend-Konfigurationsparameter

Die folgenden Parameter steuern das Auto-Tiering-Verhalten, wenn sie in der Trident-Backend-Konfiguration definiert sind:

Parameter	Erforderlich	Beschreibung
tieringPolicy	Nein	Staffelungsrichtlinie für Volumes (`auto` oder `none`)
tieringMinimumCoolingDays	Nein	Anzahl der inaktiven Tage, bevor Daten gestaffelt werden (Bereich: 2–183, Standard: 31)

Volume-Überschreibungen mithilfe von PersistentVolumeClaim-Annotationen

Unterstützte Annotationen

PersistentVolumeClaim-Annotationen ermöglichen die individuelle Überschreibung der Auto-Tiering-Einstellungen für jedes Volume.

Anmerkung	Beschreibung
`trident.netapp.io/tieringPolicy`	Überschreibt die Tiering-Richtlinie für das Volume
`trident.netapp.io/tieringMinimumCoolingDays`	Überschreibt den Wert für die Kühltage des Volumes

Beispiel: PersistentVolumeClaim mit automatischen Tiering-Überschreibungen

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: auto-tiering-pvc
  annotations:
    trident.netapp.io/tieringPolicy: auto
    trident.netapp.io/tieringMinimumCoolingDays: "45"
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: google-cloud-netapp-volumes-auto-tiering
  resources:
    requests:
      storage: 500Gi
```

Verhalten und Einschränkungen

Bereitstellungsverhalten

- Die Einstellungen für die automatische Tierierung werden nur zum Zeitpunkt der Volume-Erstellung ausgewertet und angewendet.

- Trident stimmt die Tiering-Konfiguration nach der Bereitstellung nicht ab.
- Kühltage werden ignoriert, wenn die Stufenregelung auf `none` eingestellt ist.

Plattformbeschränkungen

- Auto-tiering wird nur für NAS-Volumes (NFS und SMB) unterstützt.
- Block-Volumes (iSCSI) unterstützen kein Auto-Tiering.
- Der Google Cloud NetApp Volumes-Speicherpool muss das automatische Tiering in Google Cloud aktiviert haben.

Unterstützte Werte

- Gültiger Bereich für `tieringMinimumCoolingDays`: 2 bis 183
- Standardwert: 31

Konfigurieren Sie ein NetApp HCI- oder SolidFire-Backend

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

Element-Treiberdetails

Trident stellt den `solidfire-san` Storage-Treiber zur Kommunikation mit dem Cluster bereit. Unterstützte Zugriffsmodi sind: *ReadWriteOnce* (RWO), *ReadOnlyMany* (ROX), *ReadWriteMany* (RWX), *ReadWriteOncePod* (RWOP).

Der `solidfire-san` Speichertreiber unterstützt die Volume-Modi *file* und *block*. Für den `Filesystem` volumeMode erstellt Trident ein Volume und ein Dateisystem. Der Dateisystemtyp wird durch die StorageClass festgelegt.

Treiber	Protokoll	VolumeMode	Unterstützte Zugriffsmodi	Unterstützte Dateisysteme
`solidfire-san`	iSCSI	Block	RWO, ROX, RWX, RWOP	Kein Dateisystem. Rohes Blockgerät.
`solidfire-san`	iSCSI	Dateisystem	RWO, RWOP	`xfs`, `ext3`, `ext4`

Bevor Sie beginnen

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

- Ein unterstütztes Speichersystem, das Element-Software ausführt.
- Anmeldeinformationen für einen NetApp HCI/SolidFire-Cluster-Administrator oder Mandantenbenutzer, der Volumes verwalten kann.
- Auf allen Ihren Kubernetes-Worker-Knoten sollten die entsprechenden iSCSI-Tools installiert sein. Siehe "Informationen zur Vorbereitung der Worker-Knoten".

Backend-Konfigurationsoptionen

Siehe die folgende Tabelle für die Backend-Konfigurationsoptionen:

Parameter	Beschreibung	Standard
`version`		Immer 1
`storageDriverName`	Name des Speichertreibers	Immer „solidfire-san“
`backendName`	Benutzerdefinierter Name oder das Speicher-Backend	"solidfire_" + storage (iSCSI) IP-Adresse
`Endpoint`	MVIP für den SolidFire Cluster mit Mandantenanmeldeinformationen	
`SVIP`	Speicher (iSCSI) IP-Adresse und Port	
`labels`	Satz beliebiger, im JSON-Format vorliegender Bezeichnungen, die auf Volumes angewendet werden sollen.	""
`TenantName`	Zu verwendender Mandantenname (wird erstellt, falls nicht gefunden)	
`InitiatorIFace`	Beschränken Sie den iSCSI-Datenverkehr auf eine bestimmte Host-Schnittstelle	"default"
`UseCHAP`	Verwenden Sie CHAP zur Authentifizierung von iSCSI. Trident verwendet CHAP.	true
`AccessGroups`	Liste der zu verwendenden Access Group-IDs	Findet die ID einer Zugriffsgruppe namens "trident"
`Types`	QoS-Spezifikationen	
`limitVolumeSize`	Die Bereitstellung schlägt fehl, wenn die angeforderte Volume-Größe über diesem Wert liegt	"" (wird nicht standardmäßig erzwungen)
`debugTraceFlags`	Debug-Flags zur Verwendung bei der Fehlersuche. Beispiel, {"api":false, "method":true}	null

WARNUNG Verwenden Sie `debugTraceFlags` nicht, es sei denn, Sie führen eine Fehlerbehebung durch und benötigen einen detaillierten Protokollauszug.

Beispiel 1: Backend-Konfiguration für `solidfire-san` Treiber mit drei Volumentypen

Dieses Beispiel zeigt eine Backend-Datei mit CHAP-Authentifizierung und Modellierung von drei Volumentypen mit spezifischen QoS-Garantien. Wahrscheinlich würden Sie anschließend Speicherklassen definieren, um jede dieser mithilfe des `IOPS` storage class-Parameters zu nutzen.

```
---
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: Backend- und Speicherklassenkonfiguration für `solidfire-san`-Treiber mit virtuellen Pools

Dieses Beispiel zeigt die Backend-Definitionsdatei, die mit virtuellen Pools konfiguriert ist, zusammen mit StorageClasses, die auf sie verweisen.

Trident kopiert die im Speicherpool vorhandenen Labels beim Provisionierungsvorgang auf die Backend-Speicher-LUN. Zur Vereinfachung können Speicheradministratoren Bezeichnungen pro virtuellem Pool definieren und Volumes nach Bezeichnung gruppieren.

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

```
---
version: 1
storageDriverName: solidfire-san
Endpoint: https://<user>:<password>@<mvip>/json-rpc/8.0
SVIP: <svip>:3260
```

```
TenantName: <tenant>
UseCHAP: true
Types:
  - Type: Bronze
    Qos:
      minIOPS: 1000
      maxIOPS: 2000
      burstIOPS: 4000
  - Type: Silver
    Qos:
      minIOPS: 4000
      maxIOPS: 6000
      burstIOPS: 8000
  - Type: Gold
    Qos:
      minIOPS: 6000
      maxIOPS: 8000
      burstIOPS: 10000
type: Silver
labels:
  store: solidfire
  k8scluster: dev-1-cluster
region: us-east-1
storage:
  - labels:
      performance: gold
      cost: "4"
    zone: us-east-1a
    type: Gold
  - labels:
      performance: silver
      cost: "3"
    zone: us-east-1b
    type: Silver
  - labels:
      performance: bronze
      cost: "2"
    zone: us-east-1c
    type: Bronze
  - labels:
      performance: silver
      cost: "1"
    zone: us-east-1d
```

Die folgenden StorageClass-Definitionen beziehen sich auf die oben genannten virtuellen Pools. Mit dem `parameters.selector`-Feld gibt jede StorageClass an, welche virtuellen Pools zum Hosten eines Volumes

verwendet werden können. Das Volume weist die im gewählten virtuellen Pool definierten Aspekte auf.

Die erste StorageClass (`solidfire-gold-four`) wird dem ersten virtuellen Pool zugeordnet. Dies ist der einzige Pool, der Gold-Performance mit einer `Volume Type QoS` von Gold bietet. Die letzte StorageClass (`solidfire-silver`) bezieht sich auf jeden Speicherpool, der eine Silber-Performance bietet. Trident entscheidet, welcher virtuelle Pool ausgewählt wird und stellt sicher, dass die Speicheranforderung erfüllt wird.

```
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

- "Volumenzugriffsgruppen"

ONTAP SAN-Treiber

ONTAP SAN-Treiberübersicht

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

ONTAP SAN-Treiberdetails

Trident stellt die folgenden SAN-Speichertreiber für die Kommunikation mit dem ONTAP Cluster bereit. Unterstützte Zugriffsmodi sind: *ReadWriteOnce* (RWO), *ReadOnlyMany* (ROX), *ReadWriteMany* (RWX), *ReadWriteOncePod* (RWOP).

Treiber	Protokoll	volumeMode	Unterstützte Zugriffsmodi	Unterstützte Dateisysteme
`ontap-san`	iSCSI SCSI über FC	Block	RWO, ROX, RWX, RWOP	Kein Dateisystem; Raw-Blockgerät
`ontap-san`	iSCSI SCSI über FC	Dateisystem	RWO, RWOP ROX und RWX sind im Filesystem-Volume-Modus nicht verfügbar.	`xfs`, `ext3`, `ext4`
`ontap-san`	NVMe/TCP Siehe Weitere Überlegungen zu NVMe/TCP.	Block	RWO, ROX, RWX, RWOP	Kein Dateisystem; Raw-Blockgerät

Treiber	Protokoll	volumeMode	Unterstützte Zugriffsmodi	Unterstützte Dateisysteme
`ontap-san`	NVMe/TCP Siehe Weitere Überlegungen zu NVMe/TCP.	Dateisystem	RWO, RWOP ROX und RWX sind im Filesystem-Volume-Modus nicht verfügbar.	`xfs`, `ext3`, `ext4`
`ontap-san-economy`	iSCSI	Block	RWO, ROX, RWX, RWOP	Kein Dateisystem; Raw-Blockgerät
`ontap-san-economy`	iSCSI	Dateisystem	RWO, RWOP ROX und RWX sind im Filesystem-Volume-Modus nicht verfügbar.	`xfs`, `ext3`, `ext4`

WARNUNG

- Verwenden Sie `ontap-san-economy` nur, wenn die Anzahl der persistenten Speichernutzungen voraussichtlich höher ist als "Unterstützte ONTAP Volume-Grenzwerte".
- Verwenden Sie `ontap-nas-economy` nur, wenn die Anzahl der persistenten Speichernutzungen voraussichtlich höher ist als "Unterstützte ONTAP Volume-Grenzwerte" und der `ontap-san-economy` Treiber nicht verwendet werden kann.
- Verwenden Sie `ontap-nas-economy` nicht, wenn Sie einen Bedarf an Datenschutz, Notfallwiederherstellung oder Mobilität erwarten.
- NetApp empfiehlt nicht, Flexvol autogrow in allen ONTAP-Treibern zu verwenden, außer ontap-san. Als Workaround unterstützt Trident die Verwendung von Snapshot-Reserve und skaliert Flexvol-Volumes entsprechend.

Benutzerberechtigungen

Trident wird entweder als ONTAP- oder SVM-Administrator ausgeführt, typischerweise mit dem `admin` cluster-Benutzer oder einem `vsadmin` SVM-Benutzer oder einem Benutzer mit anderem Namen, der die gleiche Rolle hat. Für Amazon FSx for NetApp ONTAP-Bereitstellungen wird Trident entweder als ONTAP- oder SVM-Administrator ausgeführt, mit dem cluster `fsxadmin` Benutzer oder einem `vsadmin` SVM-Benutzer oder einem Benutzer mit anderem Namen, der die gleiche Rolle hat. Der `fsxadmin` Benutzer ist ein eingeschränkter Ersatz für den cluster-Admin-Benutzer.

HINWEIS

Wenn Sie den `limitAggregateUsage` Parameter verwenden, sind Cluster-Administratorrechte erforderlich. Bei Verwendung von Amazon FSx für NetApp ONTAP mit Trident wird der `limitAggregateUsage` Parameter nicht mit den `vsadmin` und `fsxadmin` Benutzerkonten funktionieren. Der Konfigurationsvorgang schlägt fehl, wenn Sie diesen Parameter angeben.

Zwar ist es möglich, in ONTAP eine restriktivere Rolle zu erstellen, die ein Trident-Treiber verwenden kann, wir raten jedoch davon ab. Die meisten neuen Versionen von Trident rufen zusätzliche APIs auf, die berücksichtigt werden müssten, was Aktualisierungen erschwert und fehleranfällig macht.

Weitere Überlegungen zu NVMe/TCP

Trident unterstützt das Non-Volatile Memory Express (NVMe)-Protokoll mit dem `ontap-san` Treiber, einschließlich:

- IPv6
- Snapshots und Klone von NVMe volumes
- Ändern der Größe eines NVMe-Volumes
- Importieren eines NVMe-Volumes, das außerhalb von Trident erstellt wurde, damit sein Lebenszyklus von Trident verwaltet werden kann
- NVMe-native Multipathing
- Geordnetes oder ungeordnetes Herunterfahren der K8s-Knoten (24.06)

Trident unterstützt nicht:

- DH-HMAC-CHAP, das nativ von NVMe unterstützt wird
- Gerätemapper (DM) Multipathing
- LUKS-Verschlüsselung

HINWEIS	NVMe wird nur mit ONTAP REST APIs unterstützt und nicht mit ONTAPI (ZAPI).

Bereiten Sie die Konfiguration des Backends mit ONTAP SAN-Treibern vor

Machen Sie sich mit den Anforderungen und Authentifizierungsoptionen für die Konfiguration eines ONTAP Backends mit ONTAP SAN-Treibern vertraut.

Anforderungen

Für alle ONTAP Backends erfordert Trident, dass mindestens ein Aggregat dem SVM zugewiesen wird.

HINWEIS	"ASA r2-Systeme" unterscheiden sich von anderen ONTAP Systemen (ASA, AFF und FAS) in der Implementierung ihrer Speicherschicht. In ASA r2-Systemen werden Speicherverfügbarkeitszonen anstelle von Aggregaten verwendet. Siehe "dies" Knowledge Base Artikel zur Zuweisung von Aggregaten zu SVMs in ASA r2-Systemen.

Beachten Sie, dass Sie auch mehrere Treiber gleichzeitig ausführen und Speicherklassen erstellen können, die auf den einen oder den anderen verweisen. Beispielsweise könnten Sie eine `san-dev` Klasse konfigurieren, die den `ontap-san` Treiber verwendet, und eine `san-default` Klasse, die den `ontap-san-economy` verwendet.

Auf allen Ihren Kubernetes-Worker-Knoten müssen die entsprechenden iSCSI-Tools installiert sein. Siehe "Bereiten Sie den Worker-Knoten vor" für Details.

Authentifizieren Sie das ONTAP Backend

Trident bietet zwei Modi zur Authentifizierung eines ONTAP Backend.

- Anmeldeinformationsbasiert: Benutzername und Passwort eines ONTAP Benutzers mit den erforderlichen Berechtigungen. Es wird empfohlen, eine vordefinierte Sicherheitsanmelderolle wie `admin` oder `vsadmin` zu verwenden, um maximale Kompatibilität mit ONTAP Versionen zu gewährleisten.

- Zertifikatsbasiert: Trident kann auch mit einem ONTAP Cluster über ein auf dem Backend installiertes Zertifikat kommunizieren. Hier muss die Backend-Definition Base64-kodierte Werte des Client-Zertifikats, des Schlüssels und, falls verwendet (empfohlen), des vertrauenswürdigen CA-Zertifikats enthalten.

Sie können bestehende Backends aktualisieren, um zwischen anmeldeinformationsbasierten und zertifikatsbasierten Authentifizierungsverfahren zu wechseln. Es wird jedoch jeweils nur ein Authentifizierungsverfahren unterstützt. Um zu einem anderen Authentifizierungsverfahren zu wechseln, müssen Sie das bestehende Verfahren aus der Backend-Konfiguration entfernen.

WARNUNG	Wenn Sie versuchen, **sowohl Anmeldeinformationen als auch Zertifikate** anzugeben, schlägt die Backend-Erstellung mit der Fehlermeldung fehl, dass mehr als ein Authentifizierungsverfahren in der Konfigurationsdatei angegeben wurde.

Aktivieren Sie die anmeldeinformationsbasierte Authentifizierung

Trident benötigt die Anmeldeinformationen eines Administrators mit SVM- oder Cluster-Berechtigungen, um mit dem ONTAP Backend zu kommunizieren. Es wird empfohlen, vordefinierte Standardrollen wie `admin` oder `vsadmin` zu verwenden. Dies gewährleistet die Vorwärtskompatibilität mit zukünftigen ONTAP Versionen, die möglicherweise Feature-APIs für zukünftige Trident Versionen bereitstellen. Eine benutzerdefinierte Sicherheits-Anmelderolle kann erstellt und mit Trident verwendet werden, wird jedoch nicht empfohlen.

Eine beispielhafte Backend-Definition sieht folgendermaßen aus:

YAML

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

JSON

```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 Zugangsdaten im Klartext

gespeichert werden. Nach der Backend-Erstellung werden Benutzernamen/Passwörter mit Base64 kodiert und als Kubernetes-Secrets gespeichert. Die Erstellung oder Aktualisierung eines Backends ist der einzige Schritt, der Kenntnisse der Zugangsdaten erfordert. Daher handelt es sich um eine ausschließlich vom Kubernetes-/Speicheradministrator durchzuführende Operation.

Aktivieren Sie die zertifikatsbasierte Authentifizierung

Neue und bestehende Backends können ein Zertifikat verwenden und mit dem ONTAP Backend kommunizieren. In der Backend-Definition sind drei Parameter erforderlich.

- clientCertificate: Base64-kodierter Wert des Client-Zertifikats.
- clientPrivateKey: Base64-kodierter Wert des zugehörigen privaten Schlüssels.
- trustedCACertificate: Base64-kodierter Wert des Zertifikats einer vertrauenswürdigen CA. Wenn eine vertrauenswürdige CA verwendet wird, muss dieser Parameter angegeben werden. Dies kann ignoriert werden, wenn keine vertrauenswürdige CA verwendet wird.

Ein typischer Arbeitsablauf umfasst die folgenden Schritte.

Schritte

1. Generieren Sie ein Clientzertifikat und einen Schlüssel. Legen Sie beim Generieren den Common Name (CN) auf den ONTAP Benutzer fest, als den Sie sich authentifizieren möchten.

```
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 wurde möglicherweise bereits vom Storage-Administrator erledigt. Ignorieren Sie dies, falls 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 Clientzertifikat und den Schlüssel (aus 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
```

HINWEIS	Nach Ausführung dieses Befehls fordert ONTAP die Eingabe des Zertifikats an. Fügen Sie den Inhalt der `k8senv.pem` Datei ein, die in Schritt 1 generiert wurde, und drücken Sie `END`, um die Installation abzuschließen.

4. Bestätigen Sie, dass die ONTAP Sicherheitsanmeldungsrolle das `cert` Authentifizierungsverfahren

unterstützt.

```
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 mit dem generierten Zertifikat. Ersetzen Sie <ONTAP Management LIF> und <vserver name> durch die Management-LIF-IP und den SVM-Namen.

```
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. Zertifikat, Schlüssel und vertrauenswürdiges CA-Zertifikat mit Base64 kodieren.

```
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 unter Verwendung der im vorherigen Schritt erhaltenen Werte.

```
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 Authentifizierungsverfahren oder rotieren Sie die Anmeldeinformationen

Sie können ein bestehendes Backend aktualisieren, um ein anderes Authentifizierungsverfahren zu verwenden oder um die Anmeldeinformationen zu rotieren. Dies funktioniert in beide Richtungen: Backends, die Benutzername/Passwort verwenden, können auf Zertifikate umgestellt werden; Backends, die Zertifikate verwenden, können auf Benutzername/Passwort umgestellt werden. Dazu müssen Sie das bestehende Authentifizierungsverfahren entfernen und das neue Authentifizierungsverfahren hinzufügen. Verwenden Sie dann die aktualisierte backend.json-Datei mit den erforderlichen Parametern, um `tridentctl backend update` auszuführen.

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

HINWEIS	Beim Rotieren von Passwörtern muss der Speicheradministrator zunächst das Passwort für den Benutzer auf ONTAP aktualisieren. Danach erfolgt ein Backend-Update. Beim Rotieren von Zertifikaten können dem Benutzer mehrere Zertifikate hinzugefügt werden. Das Backend wird dann aktualisiert, um das neue Zertifikat zu verwenden, wonach das alte Zertifikat aus dem ONTAP Cluster gelöscht werden kann.

Die Aktualisierung des Backends beeinträchtigt weder den Zugriff auf bereits erstellte Volumes noch später hergestellte Volume-Verbindungen. Eine erfolgreiche Backend-Aktualisierung zeigt an, dass Trident mit dem ONTAP Backend kommunizieren und zukünftige Volume-Operationen ausführen kann.

Erstellen einer benutzerdefinierten ONTAP Rolle für Trident

Sie können eine ONTAP-Clusterrolle mit minimalen Berechtigungen erstellen, sodass Sie nicht die ONTAP-Admin-Rolle verwenden müssen, um Vorgänge in Trident durchzuführen. Wenn Sie den Benutzernamen in einer Trident-Backend-Konfiguration angeben, verwendet Trident die von Ihnen erstellte ONTAP-Clusterrolle, um die Vorgänge auszuführen.

Weitere Informationen zum Erstellen benutzerdefinierter Trident-Rollen finden Sie unter "Trident Custom-Role-Generator".

Verwendung der ONTAP-Befehlszeile

1. Erstellen Sie eine neue Rolle mit folgendem Befehl:

```
security login role create <role_name\> -cmddirname "command" -access all
–vserver <svm_name\>
```

2. Erstellen Sie einen Benutzernamen für den Trident Benutzer:

```
security login create -username <user_name\> -application ontapi
-authmethod <password\> -role <name_of_role_in_step_1\> –vserver
<svm_name\> -comment "user_description"
```

3. Ordnen Sie die Rolle dem Benutzer zu:

```
security login modify username <user_name\> –vserver <svm_name\> -role
<role_name\> -application ontapi -application console -authmethod
<password\>
```

Verwenden von System Manager

Führen Sie die folgenden Schritte im ONTAP System Manager aus:

1. **Erstellen Sie eine benutzerdefinierte Rolle**:
   a. Um eine benutzerdefinierte Rolle auf Clusterebene zu erstellen, wählen Sie **Cluster > Settings**.

      (Oder) Um eine benutzerdefinierte Rolle auf SVM-Ebene zu erstellen, wählen Sie **Storage > Storage VMs > `required SVM`> Settings > Users and Roles**.

   b. Wählen Sie das Pfeilsymbol (→) neben **Users and Roles** aus.
   c. Wählen Sie unter **Rollen +Add** aus.
   d. Definieren Sie die Regeln für die Rolle und klicken Sie auf **Save**.
2. **Rolle dem Trident-Benutzer zuordnen**: + Führen Sie die folgenden Schritte auf der Seite **Benutzer und Rollen** aus:
   a. Wählen Sie das Symbol **+** unter **Benutzer** aus.
   b. Wählen Sie den gewünschten Benutzernamen aus und wählen Sie eine Rolle im Dropdown-Menü für **Rolle**.
   c. Klicken Sie auf **Speichern**.

Weitere Informationen finden Sie auf den folgenden Seiten:

- "Benutzerdefinierte Rollen für die Administration von ONTAP" oder "Benutzerdefinierte Rollen definieren"
- "Mit Rollen und Benutzern arbeiten"

Verbindungen mit bidirektionalem CHAP authentifizieren

Trident kann iSCSI-Sitzungen mit bidirektionalem CHAP für die `ontap-san` und `ontap-san-economy` Treiber authentifizieren. Dazu muss die `useCHAP` Option in Ihrer Backend-Definition aktiviert werden. Wenn auf `true` gesetzt, konfiguriert Trident die Standard-Initiator-Sicherheit der SVM auf bidirektionales CHAP und legt den Benutzernamen sowie die Geheimnisse aus der Backend-Datei fest. NetApp empfiehlt die

Verwendung von bidirektionalem CHAP zur Authentifizierung von Verbindungen. Siehe die folgende Beispielkonfiguration:

```
---
version: 1
storageDriverName: ontap-san
backendName: ontap_san_chap
managementLIF: 192.168.0.135
svm: ontap_iscsi_svm
useCHAP: true
username: vsadmin
password: password
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSd6cNwxyz
```

WARNUNG	Der `useCHAP` Parameter ist eine boolesche Option, die nur einmal konfiguriert werden kann. Standardmäßig ist sie auf false gesetzt. Nachdem Sie sie auf true gesetzt haben, können Sie sie nicht mehr auf false zurücksetzen.

Zusätzlich zu `useCHAP=true`, den `chapInitiatorSecret`, `chapTargetInitiatorSecret`, `chapTargetUsername` und `chapUsername` Feldern müssen diese in der Backend-Definition enthalten sein. Die Geheimnisse können nach der Erstellung eines Backends durch Ausführen von `tridentctl update` geändert werden.

So funktioniert es

Durch das Setzen von `useCHAP` auf true weist der Speicheradministrator Trident an, CHAP auf dem Storage-Backend zu konfigurieren. Dies umfasst Folgendes:

- CHAP auf der SVM einrichten:
  - Wenn der Standard-Initiator-Sicherheitstyp der SVM „none“ ist (Standardeinstellung) **und** keine bereits vorhandenen LUNs im Volume vorhanden sind, setzt Trident den Standard-Sicherheitstyp auf `CHAP` und fährt mit der Konfiguration des CHAP-Initiators sowie des Zielbenutzernamens und der Geheimnisse fort.
  - Wenn die SVM LUNs enthält, aktiviert Trident CHAP nicht auf der SVM. Dadurch wird sichergestellt, dass der Zugriff auf bereits vorhandene LUNs auf der SVM nicht eingeschränkt wird.
- Konfiguration des CHAP-Initiators und des Zielbenutzernamens sowie der Geheimnisse; diese Optionen müssen in der Backend-Konfiguration angegeben werden (wie oben gezeigt).

Nachdem das Backend erstellt wurde, erstellt Trident eine entsprechende `tridentbackend` CRD und speichert die CHAP-Secrets und Benutzernamen als Kubernetes-Secrets. Alle von Trident auf diesem Backend erstellten PVs werden über CHAP eingebunden und verbunden.

Anmeldeinformationen rotieren und Backends aktualisieren

Sie können die CHAP-Zugangsdaten aktualisieren, indem Sie die CHAP-Parameter in der `backend.json`

Datei aktualisieren. Dies erfordert die Aktualisierung der CHAP-Geheimnisse und die Verwendung des `tridentctl update` Befehls, um diese Änderungen widerzuspiegeln.

WARNUNG	Beim Aktualisieren der CHAP-Geheimnisse für ein Backend müssen Sie `tridentctl` verwenden, um das Backend zu aktualisieren. Aktualisieren Sie die Anmeldeinformationen auf dem Storage-Cluster nicht mit der ONTAP CLI oder dem ONTAP System Manager, da Trident diese Änderungen nicht übernehmen kann.

```
cat backend-san.json
{
    "version": 1,
    "storageDriverName": "ontap-san",
    "backendName": "ontap_san_chap",
    "managementLIF": "192.168.0.135",
    "svm": "ontap_iscsi_svm",
    "useCHAP": true,
    "username": "vsadmin",
    "password": "password",
    "chapInitiatorSecret": "cl9qxUpDaTeD",
    "chapTargetInitiatorSecret": "rqxigXgkeUpDaTeD",
    "chapTargetUsername": "iJF4heBRT0TCwxyz",
    "chapUsername": "uh2aNCLSd6cNwxyz",
}

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

Bestehende Verbindungen bleiben unberührt; sie bleiben aktiv, wenn die Anmeldeinformationen von Trident auf der SVM aktualisiert werden. Neue Verbindungen verwenden die aktualisierten Anmeldeinformationen und bestehende Verbindungen bleiben weiterhin aktiv. Das Trennen und erneute Verbinden alter PVs führt dazu, dass sie die aktualisierten Anmeldeinformationen verwenden.

ONTAP SAN-Konfigurationsoptionen und Beispiele

Erfahren Sie, wie Sie ONTAP SAN-Treiber mit Ihrer Trident-Installation erstellen und verwenden. Dieser Abschnitt enthält Beispiele für die Backend-Konfiguration und Details zur Zuordnung von Backends zu StorageClasses. "ASA r2-Systeme" unterscheiden sich von anderen ONTAP Systemen (ASA, AFF und FAS) in der Implementierung ihrer

Speicherschicht. Diese Unterschiede wirken sich wie angegeben auf die Verwendung bestimmter Parameter aus. "Erfahren Sie mehr über die Unterschiede zwischen ASA r2-Systemen und anderen ONTAP Systemen". In der Trident-Backend-Konfiguration müssen Sie nicht angeben, dass Ihr System ASA r2 ist. Wenn Sie `ontap-san` als `storageDriverName` auswählen, erkennt Trident automatisch das ASA r2- oder andere ONTAP-Systeme. Einige Backend-Konfigurationsparameter sind für ASA r2-Systeme, wie in der untenstehenden Tabelle angegeben, nicht anwendbar.

HINWEIS Nur der `ontap-san` driver (mit iSCSI-, NVMe/TCP- und FC-Protokollen) wird für ASA r2-Systeme unterstützt.

Backend-Konfigurationsoptionen

Siehe die folgende Tabelle für die Backend-Konfigurationsoptionen:

Parameter	Beschreibung	Standard
`version`		Immer 1
`storageDriverName`	Name des Speichertreibers	`ontap-san` oder `ontap-san-economy`
`backendName`	Benutzerdefinierter Name oder das Speicher-Backend	Treibername + "_" + dataLIF
`managementLIF`	IP-Adresse einer Cluster- oder SVM-Management-LIF. Es kann ein vollqualifizierter Domain-Name (FQDN) angegeben werden. Kann so eingestellt werden, dass IPv6-Adressen verwendet werden, wenn Trident mit dem IPv6-Flag installiert wurde. IPv6-Adressen müssen in eckigen Klammern angegeben werden, wie `[28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]`. Für einen reibungslosen MetroCluster-Switchover siehe MetroCluster Beispiel. **HINWEIS** Wenn Sie „vsadmin“-Anmeldeinformationen verwenden, `managementLIF` muss dies die der SVM sein; wenn Sie „admin“-Anmeldeinformationen verwenden, `managementLIF` muss dies die des Clusters sein.	"10.0.0.1", "[2001:1234:abcd::fefe]"

Parameter	Beschreibung	Standard
`dataLIF`	IP-Adresse des Protokoll-LIF. Kann so eingestellt werden, dass IPv6-Adressen verwendet werden, wenn Trident mit dem IPv6-Flag installiert wurde. IPv6-Adressen müssen in eckigen Klammern angegeben werden, wie `[28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]`. **Nicht für iSCSI angeben.** Trident verwendet "ONTAP Selective LUN Map" zur Ermittlung der iSCSI-LIFs, die für den Aufbau einer Multipath-Sitzung benötigt werden. Eine Warnung wird generiert, wenn `dataLIF` explizit definiert ist. **Für MetroCluster weglassen.** Siehe MetroCluster Beispiel.	Abgeleitet von der SVM
`svm`	Zu verwendende virtuelle Speichermaschine **Für MetroCluster auslassen.** Siehe MetroCluster Beispiel.	Wird abgeleitet, wenn eine SVM `managementLIF` angegeben ist
`useCHAP`	Verwenden Sie CHAP zur Authentifizierung von iSCSI für ONTAP SAN-Treiber [Boolescher Wert]. Setzen Sie auf `true`, damit Trident bidirektionales CHAP als Standardauthentifizierung für die im Backend angegebene SVM konfiguriert und verwendet. Siehe "Bereiten Sie die Konfiguration des Backends mit ONTAP SAN-Treibern vor" für Details. **Nicht unterstützt für FCP oder NVMe/TCP.**	`false`
`chapInitiatorSecret`	CHAP-Initiatorgeheimnis. Erforderlich, wenn `useCHAP=true`	""
`labels`	Satz beliebiger JSON-formatierter Bezeichnungen, die auf Volumes angewendet werden sollen	""
`chapTargetInitiatorSecret`	CHAP-Zielinitiatorgeheimnis. Erforderlich, wenn `useCHAP=true`	""
`chapUsername`	Benutzername für eingehende Anfragen. Erforderlich, wenn `useCHAP=true`	""
`chapTargetUsername`	Zielbenutzername. Erforderlich, wenn `useCHAP=true`	""
`clientCertificate`	Base64-kodierter Wert des Clientzertifikats. Wird für zertifikatbasierte Authentifizierung verwendet	""
`clientPrivateKey`	Base64-kodierter Wert des privaten Client-Schlüssels. Wird für zertifikatbasierte Authentifizierung verwendet	""
`trustedCACertificate`	Base64-kodierter Wert des vertrauenswürdigen CA-Zertifikats. Optional. Wird für zertifikatbasierte Authentifizierung verwendet.	""

Parameter	Beschreibung	Standard
`username`	Benutzername, der für die Kommunikation mit dem ONTAP Cluster benötigt wird. Wird für die anmeldeinformationsbasierte Authentifizierung verwendet. Für die Active Directory Authentifizierung siehe "Authentifizieren Sie Trident bei einer Backend-SVM mit Active Directory-Anmeldeinformationen".	""
`password`	Passwort erforderlich, um mit dem ONTAP Cluster zu kommunizieren. Wird für die anmeldeinformationsbasierte Authentifizierung verwendet. Für die Active Directory Authentifizierung siehe "Authentifizieren Sie Trident bei einer Backend-SVM mit Active Directory-Anmeldeinformationen".	""
`svm`	Zu verwendende Storage Virtual Machine	Wird abgeleitet, wenn eine SVM `managementLIF` angegeben ist
`storagePrefix`	Präfix, das beim Bereitstellen neuer Volumes in der SVM verwendet wird. Kann später nicht geändert werden. Um diesen Parameter zu aktualisieren, müssen Sie ein neues Backend erstellen.	`trident`
`aggregate`	Aggregat für die Bereitstellung (optional; falls festgelegt, muss es der SVM zugewiesen werden). Für den `ontap-nas-flexgroup` Treiber wird diese Option ignoriert. Wenn nicht zugewiesen, kann eines der verfügbaren Aggregate zur Bereitstellung eines FlexGroup Volumes verwendet werden. **HINWEIS** Wird das Aggregat in SVM aktualisiert, wird es in Trident automatisch durch Abfragen von SVM aktualisiert, ohne dass der Trident Controller neu gestartet werden muss. Wenn Sie in Trident ein bestimmtes Aggregat für die Bereitstellung von Volumes konfiguriert haben und dieses Aggregat umbenannt oder aus der SVM verschoben wird, wechselt das Backend in Trident beim Abfragen des SVM-Aggregats in den Fehlerzustand. Sie müssen entweder das Aggregat auf eines ändern, das auf der SVM vorhanden ist, oder es vollständig entfernen, um das Backend wieder online zu bringen. **Für ASA r2 Systeme nicht angeben**.	""

Parameter	Beschreibung	Standard
`limitAggregateUsage`	Die Bereitstellung schlägt fehl, wenn die Nutzung diesen Prozentsatz überschreitet. Wenn Sie ein Amazon FSx for NetApp ONTAP-Backend verwenden, geben Sie `limitAggregateUsage` nicht an. Die bereitgestellten `fsxadmin` und `vsadmin` enthalten nicht die erforderlichen Berechtigungen, um die aggregierte Nutzung abzurufen und sie mit Trident zu begrenzen. **Für ASA r2 Systeme nicht angeben**.	"" (wird nicht standardmäßig erzwungen)
`limitVolumeSize`	Die Bereitstellung schlägt fehl, wenn die angeforderte Volume-Größe diesen Wert überschreitet. Außerdem wird die maximale Größe der von ihr für LUNs verwalteten Volumes beschränkt.	"" (wird nicht standardmäßig erzwungen)
`lunsPerFlexvol`	Maximale LUNs pro FlexVol, muss im Bereich [50, 200] liegen	`100`
`debugTraceFlags`	Debug-Flags zur Verwendung bei der Fehlersuche. Beispiel, {"api":false, "method":true} nicht verwenden, es sei denn, Sie führen eine Fehlersuche durch und benötigen eine detaillierte Protokollausgabe.	`null`

Parameter	Beschreibung	Standard
`useREST`	Boolescher Parameter zur Verwendung von ONTAP REST APIs. `` `useREST` Wenn auf `true` gesetzt, verwendet Trident ONTAP REST APIs zur Kommunikation mit dem Backend; wenn auf `false` gesetzt, verwendet Trident ONTAPI (ZAPI)-Aufrufe zur Kommunikation mit dem Backend. Diese Funktion erfordert ONTAP 9.11.1 und höher. Zusätzlich muss die verwendete ONTAP-Anmelderolle Zugriff auf die `ontapi` Anwendung haben. Dies wird durch die vordefinierten `vsadmin` und `cluster-admin` Rollen erfüllt. Ab der Trident 24.06-Version und ONTAP 9.15.1 oder höher ist `useREST` standardmäßig auf `true` gesetzt; ändern Sie `useREST` auf `false`, um ONTAPI (ZAPI)-Aufrufe zu verwenden.`` `useREST` ist vollqualifiziert für NVMe/TCP. **HINWEIS** NVMe wird nur mit ONTAP REST APIs unterstützt und nicht mit ONTAPI (ZAPI). **Falls angegeben, immer auf `true` für ASA r2-Systeme setzen**.	`true` für ONTAP 9.15.1 oder höher, andernfalls `false`.
`sanType`	Verwenden Sie zum Auswählen `iscsi` für iSCSI, `nvme` für NVMe/TCP oder `fcp` für SCSI über Fibre Channel (FC).	`iscsi` falls leer

Parameter	Beschreibung	Standard
`formatOptions`	Verwenden Sie `formatOptions` , um Befehlszeilenargumente für den `mkfs` Befehl anzugeben, die bei jeder Formatierung eines Volumes angewendet werden. So können Sie das Volume nach Ihren Wünschen formatieren. Stellen Sie sicher, dass Sie die formatOptions ähnlich wie die Optionen des mkfs-Befehls angeben, jedoch ohne den Gerätepfad. Beispiel: "-E nodiscard" **Unterstützt für `ontap-san` und `ontap-san-economy` Treiber mit iSCSI-Protokoll. Zusätzlich unterstützt für ASA r2 Systeme bei Verwendung der Protokolle iSCSI und NVMe/TCP.**	
`limitVolumePoolSize`	Maximal anforderbare FlexVol-Größe bei Verwendung von LUNs im ontap-san-economy Backend.	"" (wird nicht standardmäßig erzwungen)
`denyNewVolumePools`	Beschränkt `ontap-san-economy` Backends darauf, neue FlexVol Volumes zur Aufnahme ihrer LUNs zu erstellen. Nur bereits vorhandene Flexvols werden für die Bereitstellung neuer PVs verwendet.	

Empfehlungen zur Verwendung von formatOptions

Trident empfiehlt die folgenden Optionen, um den Formatierungsprozess zu beschleunigen:

- **-E nodiscard (ext3, ext4):** Versuche nicht, Blöcke während der Erstellung des Dateisystems (mkfs) zu verwerfen (das anfängliche Verwerfen von Blöcken ist auf Solid-State-Geräten und dünnbesetzten/dünnprovisionierten Speichern sinnvoll). Dies ersetzt die veraltete Option "-K" und ist für ext3- und ext4-Dateisysteme anwendbar.
- **-K (xfs):** Versuche nicht, Blöcke beim mkfs zu verwerfen. Diese Option ist für das xfs-Dateisystem anwendbar.

Authentifizieren Sie Trident bei einer Backend-SVM mit Active Directory-Anmeldeinformationen

Sie können Trident so konfigurieren, dass es sich mit Active Directory (AD)-Anmeldeinformationen an einer Backend-SVM authentifiziert. Bevor ein AD-Konto auf die SVM zugreifen kann, müssen Sie den Zugriff des AD-Domänencontrollers auf den Cluster oder die SVM konfigurieren. Für die Clusterverwaltung mit einem AD-Konto müssen Sie einen Domänentunnel erstellen. Siehe "Konfigurieren des Zugriffs auf Active Directory-Domänencontroller in ONTAP" für Details.

Schritte

1. Konfigurieren Sie die Domain Name System (DNS)-Einstellungen für eine Backend-SVM:

```
vserver services dns create -vserver <svm_name> -dns-servers
<dns_server_ip1>,<dns_server_ip2>
```

2. Führen Sie den folgenden Befehl aus, um ein Computerkonto für die SVM in Active Directory zu erstellen:

```
vserver active-directory create -vserver DataSVM -account-name ADSERVER1
-domain demo.netapp.com
```

3. Verwenden Sie diesen Befehl, um einen AD-Benutzer oder eine Gruppe zu erstellen, die das Cluster oder die SVM verwalten.

```
security login create -vserver <svm_name> -user-or-group-name
<ad_user_or_group> -application <application> -authentication-method domain
-role vsadmin
```

4. In der Trident-Backend-Konfigurationsdatei setzen Sie die `username` und `password` Parameter auf den AD-Benutzer- bzw. Gruppennamen und das Passwort.

Backend-Konfigurationsoptionen für die Bereitstellung von Volumes

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

Parameter	Beschreibung	Standard
`spaceAllocation`	Speicherplatzzuweisung für LUNs	"true" **Falls angegeben, auf `true` für ASA r2-Systeme setzen**.
`spaceReserve`	Speicherplatzreservierungsmodus: „none“ (dünn) oder „volume“ (dick). **Auf `none` für ASA r2-Systeme einstellen.**	"none"
`snapshotPolicy`	Zu verwendende Snapshot-Richtlinie. **Für `none` ASA r2-Systeme festlegen**.	"none"
`qosPolicy`	QoS-Richtliniengruppe, die für erstellte Volumes zugewiesen werden soll. Wählen Sie eine von qosPolicy oder adaptiveQosPolicy pro Storage-Pool/Backend. Die Verwendung von QoS-Richtliniengruppen mit Trident erfordert ONTAP 9.8 oder höher. Sie sollten eine nicht gemeinsam genutzte QoS-Richtliniengruppe verwenden und sicherstellen, dass die Richtliniengruppe auf jedes einzelne Element angewendet wird. Eine gemeinsam genutzte QoS-Richtliniengruppe erzwingt die Obergrenze für den Gesamtdurchsatz aller Workloads.	""
`adaptiveQosPolicy`	Adaptive QoS-Richtliniengruppe zur Zuweisung für erstellte Volumes. Wählen Sie eine der qosPolicy oder adaptiveQosPolicy pro Speicherpool/Backend aus.	""
`snapshotReserve`	Prozentsatz des für Snapshots reservierten Speichervolumens. **Für ASA r2 Systeme nicht angeben**.	"0", falls `snapshotPolicy`"none", andernfalls ""
`splitOnClone`	Trennen Sie einen Klon bei seiner Erstellung von seinem übergeordneten Objekt	"false"

Parameter	Beschreibung	Standard
`encryption`	Aktivieren Sie NetApp Volume Encryption (NVE) auf dem neuen Volume; Standard ist `false`. NVE muss auf dem Cluster lizenziert und aktiviert sein, um diese Option zu verwenden. Wenn NAE im Backend aktiviert ist, wird jedes in Trident bereitgestellte Volume NAE-aktiviert. Weitere Informationen finden Sie unter: "Wie Trident mit NVE und NAE zusammenarbeitet".	"false" **Falls angegeben, auf `true` für ASA r2-Systeme setzen**.
`luksEncryption`	LUKS-Verschlüsselung aktivieren. Siehe "Verwenden Sie Linux Unified Key Setup (LUKS)".	"" **Auf `false` für ASA r2-Systeme setzen**.
`tieringPolicy`	Tiering-Richtlinie auf "keine" setzen **Für ASA r2-Systeme nicht angeben**.	
`nameTemplate`	Vorlage zum Erstellen benutzerdefinierter Volume-Namen.	""

Beispiele für die Volume-Bereitstellung

Hier ist ein Beispiel mit definierten Standardwerten:

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

HINWEIS	Für alle Volumes, die mit dem `ontap-san` Treiber erstellt wurden, fügt Trident dem FlexVol zusätzlich 10 Prozent Kapazität hinzu, um die LUN-Metadaten aufzunehmen. Die LUN wird mit der exakten Größe bereitgestellt, die der Benutzer im PVC anfordert. Trident fügt dem FlexVol 10 Prozent hinzu (wird als verfügbare Größe in ONTAP angezeigt). Benutzer erhalten nun die von ihnen angeforderte nutzbare Kapazität. Diese Änderung verhindert außerdem, dass LUNs schreibgeschützt werden, es sei denn, der verfügbare Speicherplatz ist vollständig genutzt. Dies gilt nicht für ontap-san-economy.

Für Backends, die `snapshotReserve` definieren, berechnet Trident die Größe der Volumes wie folgt:

```
Total volume size = [(PVC requested size) / (1 - (snapshotReserve
percentage) / 100)] * 1.1
```

Die 1,1 ist die zusätzlichen 10 Prozent, die Trident dem FlexVol hinzufügt, um die LUN-Metadaten aufzunehmen. Für `snapshotReserve` = 5 % und eine PVC-Anforderung von 5 GiB beträgt die Gesamtgröße des Volumes 5,79 GiB und die verfügbare Größe 5,5 GiB. Der `volume show`-Befehl sollte Ergebnisse ähnlich diesem Beispiel anzeigen:

Aktuell ist die Größenänderung die einzige Möglichkeit, die neue Berechnung für ein bestehendes Volume zu nutzen.

Minimale Konfigurationsbeispiele

Die folgenden Beispiele zeigen Basiskonfigurationen, bei denen die meisten Parameter auf Standardwerte eingestellt sind. Dies ist die einfachste Methode, ein Backend zu definieren.

HINWEIS	Wenn Sie Amazon FSx auf NetApp ONTAP mit Trident verwenden, empfiehlt NetApp, dass Sie für LIFs DNS-Namen anstelle von IP-Adressen angeben.

ONTAP SAN-Beispiel

Dies ist eine Basiskonfiguration mit dem `ontap-san` Treiber.

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

MetroCluster Beispiel

Sie können das Backend so konfigurieren, dass Sie die Backend-Definition nach einem Switchover und Switchback während "SVM-Replikation und -Wiederherstellung" nicht manuell aktualisieren müssen.

Für einen nahtlosen Übergang und Rückwechsel geben Sie die SVM mit `managementLIF` an und lassen Sie die `svm` Parameter weg. Beispiel:

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

ONTAP SAN economy Beispiel

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

Beispiel für zertifikatsbasierte Authentifizierung

In diesem Basiskonfigurationsbeispiel `clientCertificate`, `clientPrivateKey`, und `trustedCACertificate` (optional, wenn eine vertrauenswürdige CA verwendet wird) werden in `backend.json` gefüllt und nehmen die base64-kodierten Werte des Clientzertifikats, des privaten Schlüssels bzw. des Zertifikats der vertrauenswürdigen CA an.

```
---
version: 1
storageDriverName: ontap-san
backendName: DefaultSANBackend
managementLIF: 10.0.0.1
svm: svm_iscsi
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSd6cNwxyz
clientCertificate: ZXR0ZXJwYXB...ICMgJ3BhcGVyc2
clientPrivateKey: vciwKIyAgZG...0cnksIGRlc2NyaX
trustedCACertificate: zcyBbaG...b3Igb3duIGNsYXNz
```

Beispiele für bidirektionales CHAP

Diese Beispiele erstellen ein Backend mit `useCHAP` auf `true` gesetzt.

ONTAP SAN CHAP Beispiel

```
---
version: 1
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_iscsi
labels:
  k8scluster: test-cluster-1
  backend: testcluster1-sanbackend
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSd6cNwxyz
username: vsadmin
password: <password>
```

ONTAP SAN economy CHAP Beispiel

```
---
version: 1
storageDriverName: ontap-san-economy
managementLIF: 10.0.0.1
svm: svm_iscsi_eco
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSd6cNwxyz
username: vsadmin
password: <password>
```

NVMe/TCP-Beispiel

Sie müssen eine SVM mit NVMe auf Ihrem ONTAP Backend konfiguriert haben. Dies ist eine grundlegende Backend-Konfiguration für NVMe/TCP.

```
---
version: 1
backendName: NVMeBackend
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_nvme
username: vsadmin
password: password
sanType: nvme
useREST: true
```

SCSI über FC (FCP) Beispiel

Sie benötigen eine SVM, die mit FC auf Ihrem ONTAP Backend konfiguriert ist. Dies ist eine grundlegende Backend-Konfiguration für FC.

```
---
version: 1
backendName: fcp-backend
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_fc
username: vsadmin
password: password
sanType: fcp
useREST: true
```

Beispiel für die Backend-Konfiguration mit nameTemplate

```
---
version: 1
storageDriverName: ontap-san
backendName: ontap-san-backend
managementLIF: <ip address>
svm: svm0
username: <admin>
password: <password>
defaults:
  nameTemplate:
"{{.volume.Name}}_{{.labels.cluster}}_{{.volume.Namespace}}_{{.vo\
    lume.RequestName}}"
labels:
  cluster: ClusterA
  PVC: "{{.volume.Namespace}}_{{.volume.RequestName}}"
```

formatOptions example für den ontap-san-economy Treiber

```
---
version: 1
storageDriverName: ontap-san-economy
managementLIF: ""
svm: svm1
username: ""
password: "!"
storagePrefix: whelk_
debugTraceFlags:
  method: true
  api: true
defaults:
  formatOptions: -E nodiscard
```

Beispiele für Backends mit virtuellen Pools

In diesen Beispiel-Backend-Definitionsdateien sind spezifische Standardwerte für alle Speicherpools festgelegt, wie `spaceReserve` bei none, `spaceAllocation` bei false und `encryption` bei false. Die virtuellen Pools werden im Speicherabschnitt definiert.

Trident legt Bereitstellungsbezeichnungen im Feld „Kommentare“ fest. Kommentare werden auf dem FlexVol volume festgelegt. Trident kopiert bei der Bereitstellung alle im virtuellen Pool vorhandenen Bezeichnungen auf das Speichervolume. Zur Vereinfachung können Speicheradministratoren Bezeichnungen pro virtuellem Pool definieren und Volumes nach Bezeichnung gruppieren.

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

ONTAP SAN-Beispiel

```
---
version: 1
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_iscsi
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSd6cNwxyz
username: vsadmin
password: <password>
defaults:
  spaceAllocation: "false"
  encryption: "false"
  qosPolicy: standard
labels:
  store: san_store
  kubernetes-cluster: prod-cluster-1
region: us_east_1
storage:
  - labels:
      protection: gold
      creditpoints: "40000"
    zone: us_east_1a
    defaults:
      spaceAllocation: "true"
      encryption: "true"
      adaptiveQosPolicy: adaptive-extreme
  - labels:
      protection: silver
      creditpoints: "20000"
    zone: us_east_1b
    defaults:
      spaceAllocation: "false"
      encryption: "true"
      qosPolicy: premium
  - labels:
      protection: bronze
      creditpoints: "5000"
    zone: us_east_1c
    defaults:
      spaceAllocation: "true"
      encryption: "false"
```

ONTAP SAN economy Beispiel

```
---
version: 1
storageDriverName: ontap-san-economy
managementLIF: 10.0.0.1
svm: svm_iscsi_eco
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSd6cNwxyz
username: vsadmin
password: <password>
defaults:
  spaceAllocation: "false"
  encryption: "false"
labels:
  store: san_economy_store
region: us_east_1
storage:
  - labels:
      app: oracledb
      cost: "30"
    zone: us_east_1a
    defaults:
      spaceAllocation: "true"
      encryption: "true"
  - labels:
      app: postgresdb
      cost: "20"
    zone: us_east_1b
    defaults:
      spaceAllocation: "false"
      encryption: "true"
  - labels:
      app: mysqldb
      cost: "10"
    zone: us_east_1c
    defaults:
      spaceAllocation: "true"
      encryption: "false"
  - labels:
      department: legal
      creditpoints: "5000"
    zone: us_east_1c
```

```
    defaults:
      spaceAllocation: "true"
      encryption: "false"
```

NVMe/TCP-Beispiel

```
---
version: 1
storageDriverName: ontap-san
sanType: nvme
managementLIF: 10.0.0.1
svm: nvme_svm
username: vsadmin
password: <password>
useREST: true
defaults:
  spaceAllocation: "false"
  encryption: "true"
storage:
  - labels:
      app: testApp
      cost: "20"
    defaults:
      spaceAllocation: "false"
      encryption: "false"
```

Backends zu StorageClasses zuordnen

Die folgenden StorageClass-Definitionen beziehen sich auf die Beispiele für Backends mit virtuellen Pools. Mithilfe des `parameters.selector`-Feldes gibt jede StorageClass an, welche virtuellen Pools zum Hosten eines Volumes verwendet werden können. Das Volume weist die im gewählten virtuellen Pool definierten Aspekte auf.

- Die `protection-gold` StorageClass wird dem ersten virtuellen Pool im `ontap-san` Backend zugeordnet. Dies ist der einzige Pool, der Schutz auf Gold-Niveau bietet.

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

- Die `protection-not-gold` StorageClass wird dem zweiten und dritten virtuellen Pool im `ontap-san` Backend zugeordnet. Dies sind die einzigen Pools, die ein anderes Schutzniveau als Gold bieten.

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

- Die `app-mysqldb` StorageClass wird dem dritten virtuellen Pool im `ontap-san-economy` Backend zugeordnet. Dies ist der einzige Pool, der eine Speicherpoolkonfiguration für den Anwendungstyp mysqldb bietet.

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

- Die `protection-silver-creditpoints-20k` StorageClass wird dem zweiten virtuellen Pool im `ontap-san` Backend zugeordnet. Dies ist der einzige Pool, der Schutz auf Silber-Niveau und 20000 Kreditpunkte bietet.

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

- Die `creditpoints-5k` StorageClass wird dem dritten virtuellen Pool im `ontap-san` Backend und dem vierten virtuellen Pool im `ontap-san-economy` Backend zugeordnet. Dies sind die einzigen Poolangebote mit 5000 Kreditpunkten.

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

- Die `my-test-app-sc` StorageClass wird dem `testAPP` virtuellen Pool im `ontap-san` Treiber mit `sanType: nvme` zugeordnet. Dies ist das einzige Poolangebot `testApp`.

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

Trident entscheidet, welcher virtuelle Pool ausgewählt wird und stellt sicher, dass die Speicheranforderung erfüllt wird.

ONTAP NAS-Treiber

ONTAP NAS-Treiberübersicht

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

ONTAP NAS-Treiberdetails

Trident stellt die folgenden NAS-Speichertreiber für die Kommunikation mit dem ONTAP Cluster bereit. Unterstützte Zugriffsmodi sind: *ReadWriteOnce* (RWO), *ReadOnlyMany* (ROX), *ReadWriteMany* (RWX), *ReadWriteOncePod* (RWOP).

Treiber	Protokoll	volumeMode	Unterstützte Zugriffsmodi	Unterstützte Dateisysteme
`ontap-nas`	NFS SMB	Dateisystem	RWO, ROX, RWX, RWOP	"", `nfs`, `smb`
`ontap-nas-economy`	NFS SMB	Dateisystem	RWO, ROX, RWX, RWOP	"", `nfs`, `smb`
`ontap-nas-flexgroup`	NFS SMB	Dateisystem	RWO, ROX, RWX, RWOP	"", `nfs`, `smb`

WARNUNG

- Verwenden Sie `ontap-san-economy` nur, wenn die Anzahl der persistenten Speichernutzungen voraussichtlich höher ist als "Unterstützte ONTAP Volume-Grenzwerte".
- Verwenden Sie `ontap-nas-economy` nur, wenn die Anzahl der persistenten Speichernutzungen voraussichtlich höher ist als "Unterstützte ONTAP Volume-Grenzwerte" und der `ontap-san-economy` Treiber nicht verwendet werden kann.
- Verwenden Sie `ontap-nas-economy` nicht, wenn Sie einen Bedarf an Datenschutz, Notfallwiederherstellung oder Mobilität erwarten.
- NetApp empfiehlt nicht, Flexvol autogrow in allen ONTAP-Treibern zu verwenden, außer ontap-san. Als Workaround unterstützt Trident die Verwendung von Snapshot-Reserve und skaliert Flexvol-Volumes entsprechend.

Benutzerberechtigungen

Trident wird entweder als ONTAP- oder SVM-Administrator ausgeführt, typischerweise mit dem `admin` cluster-Benutzer oder einem `vsadmin` SVM-Benutzer oder einem Benutzer mit anderem Namen, der die gleiche Rolle hat.

Für Amazon FSx for NetApp ONTAP-Bereitstellungen wird Trident entweder als ONTAP- oder SVM-Administrator ausgeführt, mit dem cluster `fsxadmin` Benutzer oder einem `vsadmin` SVM-Benutzer oder einem Benutzer mit anderem Namen, der die gleiche Rolle hat. Der `fsxadmin` Benutzer ist ein eingeschränkter Ersatz für den cluster-Admin-Benutzer.

HINWEIS

Wenn Sie den `limitAggregateUsage` Parameter verwenden, sind Cluster-Administratorrechte erforderlich. Bei Verwendung von Amazon FSx für NetApp ONTAP mit Trident wird der `limitAggregateUsage` Parameter nicht mit den `vsadmin` und `fsxadmin` Benutzerkonten funktionieren. Der Konfigurationsvorgang schlägt fehl, wenn Sie diesen Parameter angeben.

Zwar ist es möglich, in ONTAP eine restriktivere Rolle zu erstellen, die ein Trident-Treiber verwenden kann, wir raten jedoch davon ab. Die meisten neuen Versionen von Trident rufen zusätzliche APIs auf, die berücksichtigt werden müssten, was Aktualisierungen erschwert und fehleranfällig macht.

Bereiten Sie die Konfiguration eines Backends mit ONTAP NAS-Treibern vor

Verstehen Sie die Anforderungen, Authentifizierungsoptionen und Exportrichtlinien für die Konfiguration eines ONTAP Backends mit ONTAP NAS-Treibern. Ab der Version 25.10 unterstützt NetApp Trident "NetApp AFX-Speichersystem". NetApp AFX-Speichersysteme unterscheiden sich von anderen ONTAP-Systemen (ASA, AFF und FAS) in der Implementierung ihrer Speicherschicht. In der Trident-Backend-Konfiguration müssen Sie nicht angeben, dass Ihr System AFX ist. Wenn Sie `ontap-nas` als `storageDriverName` auswählen, erkennt Trident die AFX-Systeme automatisch.

HINWEIS	Nur der `ontap-nas` Treiber (mit NFS-Protokoll) wird für AFX-Systeme unterstützt; das SMB-Protokoll wird nicht unterstützt.

Anforderungen

- Für alle ONTAP Backends erfordert Trident, dass mindestens ein Aggregat dem SVM zugewiesen wird.
- Sie können mehr als einen Treiber ausführen und Speicherklassen erstellen, die auf den einen oder den anderen verweisen. Beispielsweise könnten Sie eine Gold-Klasse konfigurieren, die den `ontap-nas` Treiber verwendet, und eine Bronze-Klasse, die den `ontap-nas-economy` verwendet.
- Auf allen Ihren Kubernetes-Worker-Knoten müssen die entsprechenden NFS-Tools installiert sein. Weitere Informationen finden Sie unter "hier,".
- Trident unterstützt SMB-Volumes nur, wenn sie in Pods eingebunden sind, die auf Windows-Knoten ausgeführt werden. Siehe Bereiten Sie die Bereitstellung von SMB-Volumes vor für Details.

Authentifizieren Sie das ONTAP Backend

Trident bietet zwei Modi zur Authentifizierung eines ONTAP Backend.

- Anmeldeinformationsbasiert: Dieser Modus erfordert ausreichende Berechtigungen für das ONTAP Backend. Es wird empfohlen, ein Konto zu verwenden, das einer vordefinierten Sicherheitsanmelderolle zugeordnet ist, wie `admin` oder `vsadmin` um maximale Kompatibilität mit ONTAP Versionen zu gewährleisten.
- Zertifikatsbasiert: In diesem Modus ist ein auf dem Backend installiertes Zertifikat erforderlich, damit Trident mit einem ONTAP Cluster kommunizieren kann. Hier muss die Backend-Definition Base64-kodierte Werte des Client-Zertifikats, des Schlüssels und, falls verwendet (empfohlen), des vertrauenswürdigen CA-Zertifikats enthalten.

Sie können bestehende Backends aktualisieren, um zwischen anmeldeinformationsbasierten und zertifikatsbasierten Authentifizierungsverfahren zu wechseln. Es wird jedoch jeweils nur ein Authentifizierungsverfahren unterstützt. Um zu einem anderen Authentifizierungsverfahren zu wechseln, müssen Sie das bestehende Verfahren aus der Backend-Konfiguration entfernen.

WARNUNG	Wenn Sie versuchen, **sowohl Anmeldeinformationen als auch Zertifikate** anzugeben, schlägt die Backend-Erstellung mit der Fehlermeldung fehl, dass mehr als ein Authentifizierungsverfahren in der Konfigurationsdatei angegeben wurde.

Aktivieren Sie die anmeldeinformationsbasierte Authentifizierung

Trident benötigt die Anmeldeinformationen eines Administrators mit SVM- oder Cluster-Berechtigungen, um mit dem ONTAP Backend zu kommunizieren. Es wird empfohlen, vordefinierte Standardrollen wie `admin` oder

`vsadmin` zu verwenden. Dies gewährleistet die Vorwärtskompatibilität mit zukünftigen ONTAP Versionen, die möglicherweise Feature-APIs für zukünftige Trident Versionen bereitstellen. Eine benutzerdefinierte Sicherheits-Anmelderolle kann erstellt und mit Trident verwendet werden, wird jedoch nicht empfohlen.

Eine beispielhafte Backend-Definition sieht folgendermaßen aus:

YAML

```yaml
---
version: 1
backendName: ExampleBackend
storageDriverName: ontap-nas
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
svm: svm_nfs
credentials:
  name: secret-backend-creds
```

JSON

```json
{
  "version": 1,
  "backendName": "ExampleBackend",
  "storageDriverName": "ontap-nas",
  "managementLIF": "10.0.0.1",
  "dataLIF": "10.0.0.2",
  "svm": "svm_nfs",
  "credentials": {
        "name": "secret-backend-creds"
    }
}
```

Beachten Sie, dass die Backend-Definition der einzige Ort ist, an dem die Zugangsdaten im Klartext gespeichert werden. Nach der Backend-Erstellung werden Benutzernamen/Passwörter mit Base64 kodiert und als Kubernetes-Secrets gespeichert. Die Erstellung/Aktualisierung eines Backends ist der einzige Schritt, der Kenntnisse der Zugangsdaten erfordert. Daher handelt es sich um eine ausschließlich vom Kubernetes-/Speicheradministrator durchzuführende Operation.

Zertifikatsbasierte Authentifizierung aktivieren

Neue und bestehende Backends können ein Zertifikat verwenden und mit dem ONTAP Backend kommunizieren. In der Backend-Definition sind drei Parameter erforderlich.

- clientCertificate: Base64-kodierter Wert des Client-Zertifikats.
- clientPrivateKey: Base64-kodierter Wert des zugehörigen privaten Schlüssels.
- trustedCACertificate: Base64-kodierter Wert des Zertifikats einer vertrauenswürdigen CA. Wenn eine

vertrauenswürdige CA verwendet wird, muss dieser Parameter angegeben werden. Dies kann ignoriert werden, wenn keine vertrauenswürdige CA verwendet wird.

Ein typischer Arbeitsablauf umfasst die folgenden Schritte.

Schritte

1. Generieren Sie ein Clientzertifikat und einen Schlüssel. Legen Sie beim Generieren den Common Name (CN) auf den ONTAP Benutzer fest, als den Sie sich authentifizieren möchten.

```
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 wurde möglicherweise bereits vom Storage-Administrator erledigt. Ignorieren Sie dies, falls 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 Clientzertifikat und den Schlüssel (aus 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 das `cert` Authentifizierungsverfahren unterstützt.

```
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 mit dem generierten Zertifikat. Ersetzen Sie <ONTAP Management LIF> und <vserver name> durch die Management-LIF-IP und den SVM-Namen. Sie müssen sicherstellen, dass die Service-Richtlinie des LIF auf `default-data-management` gesetzt ist.

```
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. Zertifikat, Schlüssel und vertrauenswürdiges CA-Zertifikat mit Base64 kodieren.

```
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 unter Verwendung der im vorherigen Schritt erhaltenen Werte.

```
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 Authentifizierungsverfahren oder rotieren Sie die Anmeldeinformationen

Sie können ein bestehendes Backend aktualisieren, um ein anderes Authentifizierungsverfahren zu verwenden oder um die Anmeldeinformationen zu rotieren. Dies funktioniert in beide Richtungen: Backends, die Benutzername/Passwort verwenden, können auf Zertifikate umgestellt werden; Backends, die Zertifikate verwenden, können auf Benutzername/Passwort umgestellt werden. Dazu müssen Sie das bestehende Authentifizierungsverfahren entfernen und das neue Authentifizierungsverfahren hinzufügen. Verwenden Sie dann die aktualisierte backend.json-Datei mit den erforderlichen Parametern, um `tridentctl update backend` auszuführen.

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

HINWEIS

Beim Rotieren von Passwörtern muss der Speicheradministrator zunächst das Passwort für den Benutzer auf ONTAP aktualisieren. Danach erfolgt ein Backend-Update. Beim Rotieren von Zertifikaten können dem Benutzer mehrere Zertifikate hinzugefügt werden. Das Backend wird dann aktualisiert, um das neue Zertifikat zu verwenden, wonach das alte Zertifikat aus dem ONTAP Cluster gelöscht werden kann.

Die Aktualisierung des Backends beeinträchtigt weder den Zugriff auf bereits erstellte Volumes noch später

hergestellte Volume-Verbindungen. Eine erfolgreiche Backend-Aktualisierung zeigt an, dass Trident mit dem ONTAP Backend kommunizieren und zukünftige Volume-Operationen ausführen kann.

Erstellen einer benutzerdefinierten ONTAP Rolle für Trident

Sie können eine ONTAP-Clusterrolle mit minimalen Berechtigungen erstellen, sodass Sie nicht die ONTAP-Admin-Rolle verwenden müssen, um Vorgänge in Trident durchzuführen. Wenn Sie den Benutzernamen in einer Trident-Backend-Konfiguration angeben, verwendet Trident die von Ihnen erstellte ONTAP-Clusterrolle, um die Vorgänge auszuführen.

Weitere Informationen zum Erstellen benutzerdefinierter Trident-Rollen finden Sie unter "Trident Custom-Role-Generator".

Verwendung der ONTAP-Befehlszeile

1. Erstellen Sie eine neue Rolle mit folgendem Befehl:

```
security login role create <role_name\> -cmddirname "command" -access all
–vserver <svm_name\>
```

2. Erstellen Sie einen Benutzernamen für den Trident Benutzer:

```
security login create -username <user_name\> -application ontapi
-authmethod <password\> -role <name_of_role_in_step_1\> -vserver
<svm_name\> -comment "user_description"
```

3. Ordnen Sie die Rolle dem Benutzer zu:

```
security login modify username <user_name\> –vserver <svm_name\> -role
<role_name\> -application ontapi -application console -authmethod
<password\>
```

Verwenden von System Manager

Führen Sie die folgenden Schritte im ONTAP System Manager aus:

1. **Erstellen Sie eine benutzerdefinierte Rolle**:
   a. Um eine benutzerdefinierte Rolle auf Clusterebene zu erstellen, wählen Sie **Cluster > Settings**.

      (Oder) Um eine benutzerdefinierte Rolle auf SVM-Ebene zu erstellen, wählen Sie **Storage > Storage VMs > `required SVM`> Settings > Users and Roles**.

   b. Wählen Sie das Pfeilsymbol (→) neben **Users and Roles** aus.
   c. Wählen Sie unter **Rollen +Add** aus.
   d. Definieren Sie die Regeln für die Rolle und klicken Sie auf **Save**.
2. **Rolle dem Trident-Benutzer zuordnen**: + Führen Sie die folgenden Schritte auf der Seite **Benutzer und Rollen** aus:
   a. Wählen Sie das Symbol **+** unter **Benutzer** aus.
   b. Wählen Sie den gewünschten Benutzernamen aus und wählen Sie eine Rolle im Dropdown-Menü für **Rolle**.
   c. Klicken Sie auf **Speichern**.

Weitere Informationen finden Sie auf den folgenden Seiten:

- "Benutzerdefinierte Rollen für die Administration von ONTAP" oder "Benutzerdefinierte Rollen definieren"
- "Mit Rollen und Benutzern arbeiten"

NFS-Exportrichtlinien verwalten

Trident verwendet NFS-Exportregeln, um den Zugriff auf die Volumes zu steuern, die es bereitstellt.

Trident bietet zwei Optionen für die Arbeit mit Exportrichtlinien:

- Trident kann die Exportregel dynamisch selbst verwalten; in diesem Betriebsmodus gibt der Speicheradministrator eine Liste von CIDR-Blöcken an, die zulässige IP-Adressen repräsentieren. Trident fügt anwendbare Knoten-IPs, die in diese Bereiche fallen, der Exportregel beim Veröffentlichen automatisch hinzu. Alternativ, wenn keine CIDRs angegeben werden, werden alle globalen Unicast-IPs, die auf dem Knoten gefunden werden, auf den das Volume veröffentlicht wird, zur Exportregel hinzugefügt.
- Speicheradministratoren können eine Exportregel erstellen und Regeln manuell hinzufügen. Trident verwendet die Standard-Exportregel, sofern in der Konfiguration kein anderer Name für die Exportregel angegeben ist.

Exportregeln dynamisch verwalten

Trident bietet die Möglichkeit, Exportregeln für ONTAP-Backends dynamisch zu verwalten. Dadurch erhält der Speicheradministrator die Möglichkeit, einen zulässigen Adressraum für Worker-Knoten-IPs anzugeben, anstatt explizite Regeln manuell zu definieren. Dies vereinfacht die Verwaltung der Exportregel erheblich; Änderungen an der Exportregel erfordern keine manuelle Intervention mehr am Storage-Cluster. Außerdem hilft dies, den Zugriff auf den Storage-Cluster nur auf Worker-Knoten zu beschränken, die Volumes einbinden und deren IPs im angegebenen Bereich liegen, was eine feingranulare und automatisierte Verwaltung unterstützt.

HINWEIS	Verwenden Sie keine Netzwerkadressübersetzung (NAT), wenn Sie dynamische Exportregeln verwenden. Mit NAT sieht der Speicherkontroller die Frontend-NAT-Adresse und nicht die tatsächliche IP-Hostadresse, sodass der Zugriff verweigert wird, wenn in den Exportregeln keine Übereinstimmung gefunden wird.

Beispiel

Es gibt zwei Konfigurationsoptionen, die verwendet werden müssen. Hier ist ein Beispiel für eine Backend-Definition:

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

HINWEIS	Bei Verwendung dieser Funktion müssen Sie sicherstellen, dass die Root-Verbindung in Ihrer SVM über eine zuvor erstellte Exportregel verfügt, die den CIDR-Block des Knotens zulässt (wie z. B. die Standard-Exportregel). Befolgen Sie stets die von NetApp empfohlenen Best Practices, um eine SVM für Trident zu dedizieren.

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

- `autoExportPolicy` ist auf `true` gesetzt. Dies bedeutet, dass Trident für jedes mit diesem Backend

bereitgestellte Volume für die `svm1` SVM eine Exportregel erstellt und das Hinzufügen sowie Löschen von Regeln mithilfe von `autoexportCIDRs` Adressblöcken verwaltet. Solange ein Volume nicht an einen Knoten angehängt ist, verwendet das Volume eine leere Exportregel ohne Regeln, um unerwünschten Zugriff auf dieses Volume zu verhindern. Wenn ein Volume auf einem Knoten veröffentlicht wird, erstellt Trident eine Exportregel mit demselben Namen wie der zugrunde liegende qtree, der die Knoten-IP innerhalb des angegebenen CIDR-Blocks enthält. Diese IPs werden auch der Exportregel hinzugefügt, die vom übergeordneten FlexVol volume verwendet wird.

  - Beispiel:
    - Backend-UUID 403b5326-8482-40db-96d0-d83fb3f4daec
    - `autoExportPolicy` eingestellt auf `true`
    - Speicherpräfix `trident`
    - PVC UUID a79bcf5f-7b6d-4a40-9876-e2551f159c1c
    - Der Qtree mit dem Namen trident_pvc_a79bcf5f_7b6d_4a40_9876_e2551f159c1c erstellt eine Exportrichtlinie für das FlexVol mit dem Namen `trident-403b5326-8482-40db96d0-d83fb3f4daec`, eine Exportrichtlinie für den Qtree mit dem Namen `trident_pvc_a79bcf5f_7b6d_4a40_9876_e2551f159c1c` und eine leere Exportrichtlinie mit dem Namen `trident_empty` auf dem SVM. Die Regeln der FlexVol-Exportrichtlinie bilden eine Obermenge aller in den Qtree-Exportrichtlinien enthaltenen Regeln. Die leere Exportrichtlinie wird von allen nicht angehängten Volumes wiederverwendet.

- `autoExportCIDRs` enthält eine Liste von Adressblöcken. Dieses Feld ist optional und hat standardmäßig den Wert ["0.0.0.0/0", "::/0"]. Falls nicht definiert, fügt Trident alle global gültigen Unicast-Adressen hinzu, die auf den Worker-Knoten mit Veröffentlichungen gefunden wurden.

In diesem Beispiel wird der `192.168.0.0/24` Adressraum bereitgestellt. Dies bedeutet, dass Kubernetes-Knoten-IPs, die in diesen Adressbereich fallen und Veröffentlichungen enthalten, der von Trident erstellten Exportregel hinzugefügt werden. Wenn Trident einen Knoten registriert, auf dem es ausgeführt wird, ruft es die IP-Adressen des Knotens ab und prüft sie anhand der in `autoExportCIDRs` bereitgestellten Adressblöcke. Zum Veröffentlichungszeitpunkt erstellt Trident nach dem Filtern der IPs die Exportregel für die Client-IPs des Knotens, auf den veröffentlicht wird.

Sie können `autoExportPolicy` und `autoExportCIDRs` für Backends nach deren Erstellung aktualisieren. Sie können neue CIDRs für ein automatisch verwaltetes Backend hinzufügen oder bestehende CIDRs löschen. Gehen Sie beim Löschen von CIDRs vorsichtig vor, um sicherzustellen, dass bestehende Verbindungen nicht getrennt werden. Sie können auch wählen, `autoExportPolicy` für ein Backend zu deaktivieren und auf eine manuell erstellte Exportrichtlinie zurückzugreifen. Dies erfordert das Setzen des `exportPolicy` Parameters in Ihrer Backend-Konfiguration.

Nachdem Trident ein Backend erstellt oder aktualisiert hat, können Sie das Backend mit `tridentctl` oder der entsprechenden `tridentbackend` CRD überprüfen:

```
./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 ein Knoten entfernt wird, überprüft Trident alle Exportregeln, um die Zugriffsregeln zu entfernen, die dem Knoten entsprechen. Durch das Entfernen dieser Knoten-IP aus den Exportregeln der verwalteten Backends verhindert Trident unerwünschte Einbindungen, es sei denn, diese IP wird von einem neuen Knoten im Cluster wiederverwendet.

Bei bereits vorhandenen Backends stellt die Aktualisierung des Backends mit `tridentctl update backend` sicher, dass Trident die Exportregeln automatisch verwaltet. Dabei werden bei Bedarf zwei neue Exportregeln erstellt, die nach der UUID und dem Qtree-Namen des Backends benannt sind. Volumes, die auf dem Backend vorhanden sind, verwenden nach dem Aushängen und erneuten Einhängen die neu erstellten Exportregeln.

HINWEIS	Das Löschen eines Backends mit automatisch verwalteten Exportregeln löscht die dynamisch erstellte Exportregel. Wenn das Backend neu erstellt wird, wird es als neues Backend behandelt und führt zur Erstellung einer neuen Exportregel.

Wird die IP-Adresse eines aktiven Knotens geändert, muss der Trident Pod auf dem Knoten neu gestartet werden. Trident aktualisiert anschließend die Exportregel für die von ihm verwalteten Backends, um diese IP-Änderung widerzuspiegeln.

Bereiten Sie die Bereitstellung von SMB-Volumes vor

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

WARNUNG	Sie müssen sowohl das NFS- als auch das SMB/CIFS-Protokoll auf der SVM konfigurieren, um ein `ontap-nas-economy` SMB-Volume für ONTAP On-Premises-Cluster zu erstellen. Wenn eines dieser Protokolle nicht konfiguriert ist, schlägt die Erstellung des SMB-Volumes fehl.

HINWEIS `autoExportPolicy` is not supported for SMB-Volumes.

Bevor Sie beginnen

Bevor Sie SMB-Volumes bereitstellen können, müssen Sie Folgendes haben.

- Ein Kubernetes-Cluster mit einem Linux-Controller-Knoten und mindestens einem Windows-Worker-Knoten, auf dem Windows Server 2022 läuft. Trident unterstützt SMB-Volumes nur, wenn sie in Pods eingebunden sind, die auf Windows-Knoten ausgeführt werden.
- Mindestens ein Trident secret, das Ihre Active Directory-Anmeldeinformationen enthält. Um ein secret zu generieren `smbcreds`:

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

- Ein als Windows-Dienst konfigurierter CSI-Proxy. Um einen `csi-proxy` zu konfigurieren, siehe "GitHub: CSI Proxy" oder "GitHub: CSI-Proxy für Windows" für Kubernetes-Knoten, die unter Windows laufen.

Schritte

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

   HINWEIS SMB-Shares sind für Amazon FSx for ONTAP erforderlich.

   Sie können die SMB-Administratorfreigaben auf zwei Arten erstellen: entweder mit dem "Microsoft Management Console" Shared Folders Snap-In oder mit der ONTAP CLI. Um die SMB-Freigaben mit der ONTAP CLI zu erstellen:

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

      Der `vserver cifs share create` Befehl prüft den in der Option -path beim Erstellen der Freigabe angegebenen Pfad. Wenn der angegebene Pfad nicht existiert, schlägt der Befehl fehl.

   b. Erstellen Sie eine SMB-Freigabe, die dem angegebenen SVM zugeordnet ist:

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

   c. Überprüfen Sie, ob die Freigabe erstellt wurde:

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

   HINWEIS Weitere Einzelheiten finden Sie in "Erstellen Sie eine SMB-Freigabe".

2. Bei der Erstellung des Backends müssen Sie Folgendes konfigurieren, um SMB-Volumes anzugeben. Alle FSx for ONTAP Backend-Konfigurationsoptionen finden Sie unter "FSx for ONTAP Konfigurationsoptionen

und Beispiele".

Parameter	Beschreibung	Beispiel
`smbShare`	Sie können eines der folgenden angeben: den Namen einer SMB-Freigabe, die mit der Microsoft Management Console oder ONTAP CLI erstellt wurde; einen Namen, damit Trident die SMB-Freigabe erstellen kann; oder Sie können den Parameter leer lassen, um den gemeinsamen Freigabezugriff auf Volumes zu verhindern. Dieser Parameter ist optional für lokale ONTAP. Dieser Parameter ist für Amazon FSx for ONTAP Backends erforderlich und darf nicht leer sein.	`smb-share`
`nasType`	**Muss auf `smb` gesetzt werden.** Wenn null, wird standardmäßig `nfs` verwendet.	`smb`
`securityStyle`	Sicherheitsstil für neue Volumes. **Muss auf `ntfs` oder `mixed` für SMB-Volumes eingestellt werden.**	`ntfs` or `mixed` für SMB-Volumes
`unixPermissions`	Modus für neue Volumes. **Muss für SMB-Volumes leer bleiben.**	""

Sichere SMB-Verbindungen aktivieren

Ab der Version 25.06 unterstützt NetApp Trident die sichere Bereitstellung von SMB-Volumes, die mit `ontap-nas` und `ontap-nas-economy` Backends erstellt wurden. Wenn Secure SMB aktiviert ist, können Sie Active Directory (AD)-Benutzern und -Benutzergruppen mithilfe von Zugriffssteuerungslisten (ACLs) kontrollierten Zugriff auf die SMB-Freigaben gewähren.

Wichtige Punkte

- Das Importieren `ontap-nas-economy` von Volumes wird nicht unterstützt.
- Es werden nur schreibgeschützte Klone für `ontap-nas-economy` volumes unterstützt.
- Wenn Secure SMB aktiviert ist, ignoriert Trident die im Backend angegebene SMB-Freigabe.
- Das Aktualisieren der PVC-Annotation, der Speicherklassenannotation und des Backend-Felds aktualisiert nicht die SMB share ACL.
- Die in der Annotation des Klon-PVC angegebene SMB-Share-ACL hat Vorrang vor denen im Quell-PVC.
- Stellen Sie sicher, dass Sie gültige AD-Benutzer angeben, wenn Sie sicheres SMB aktivieren. Ungültige Benutzer werden nicht zur ACL hinzugefügt.
- Wenn Sie dem gleichen AD-Benutzer im Backend, in der Speicherklasse und im PVC unterschiedliche Berechtigungen zuweisen, ist die Berechtigungspriorität: PVC, Speicherklasse und dann Backend.
- Secure SMB wird für `ontap-nas`managed Volume-Importe unterstützt und ist für unmanaged Volume-Importe nicht anwendbar.

Schritte

1. Geben Sie adAdminUser in TridentBackendConfig wie im folgenden Beispiel an:

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-nas
  managementLIF: 10.193.176.x
  svm: svm0
  useREST: true
  defaults:
    adAdminUser: tridentADtest
  credentials:
    name: backend-tbc-ontap-invest-secret
```

2. Fügen Sie die Annotation in die Speicherklasse ein.

   Fügen Sie die `trident.netapp.io/smbShareAdUser` Annotation zur Storage Class hinzu, um sicheres SMB ohne Fehler zu aktivieren. Der für die Annotation `trident.netapp.io/smbShareAdUser` angegebene Benutzerwert sollte derselbe sein wie der im `smbcreds` Secret angegebene Benutzername. Sie können eines der folgenden für `smbShareAdUserPermission` auswählen: `full_control`, `change` oder `read`. Die Standardberechtigung ist `full_control`.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-smb-sc
  annotations:
    trident.netapp.io/smbShareAdUserPermission: change
    trident.netapp.io/smbShareAdUser: tridentADuser
parameters:
  backendType: ontap-nas
  csi.storage.k8s.io/node-stage-secret-name: smbcreds
  csi.storage.k8s.io/node-stage-secret-namespace: trident
  trident.netapp.io/nasType: smb
provisioner: csi.trident.netapp.io
reclaimPolicy: Delete
volumeBindingMode: Immediate
```

1. Erstellen Sie ein PVC.

   Das folgende Beispiel erstellt eine PVC:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-pvc4
  namespace: trident
  annotations:
    trident.netapp.io/snapshotDirectory: "true"
    trident.netapp.io/smbShareAccessControl: |
      read:
        - tridentADtest
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: ontap-smb-sc
```

ONTAP NAS-Konfigurationsoptionen und Beispiele

Lernen Sie, wie Sie ONTAP NAS-Treiber mit Ihrer Trident-Installation erstellen und verwenden. Dieser Abschnitt enthält Beispiele für die Backend-Konfiguration und Details zur Zuordnung von Backends zu StorageClasses. Ab der Version 25.10 unterstützt NetApp Trident "NetApp AFX-Speichersysteme". NetApp AFX-Speichersysteme unterscheiden sich von anderen ONTAP-basierten Systemen (ASA, AFF und FAS) in der Implementierung ihrer Speicherschicht.

HINWEIS	Nur der `ontap-nas` Treiber (mit NFS-Protokoll) wird für NetApp AFX-Systeme unterstützt; das SMB-Protokoll wird nicht unterstützt.

Backend-Konfigurationsoptionen

In der Trident-Backend-Konfiguration müssen Sie nicht angeben, dass Ihr System ein NetApp AFX-Speichersystem ist. Wenn Sie `ontap-nas` als `storageDriverName` auswählen, erkennt Trident das AFX-Speichersystem automatisch. Einige Backend-Konfigurationsparameter sind für AFX-Speichersysteme nicht anwendbar.

Die folgende Tabelle zeigt die Backend-Konfigurationsoptionen:

Parameter	Beschreibung	Standard
`version`		Immer 1

Parameter	Beschreibung	Standard
`storageDriverName`	Name des Speichertreibers **HINWEIS** Für NetApp AFX-Systeme wird nur `ontap-nas` unterstützt.	`ontap-nas`, `ontap-nas-economy`, oder `ontap-nas-flexgroup`
`backendName`	Benutzerdefinierter Name oder das Speicher-Backend	Treibername + "_" + dataLIF
`managementLIF`	IP-Adresse eines Clusters oder einer SVM-Management-LIF. Ein vollqualifizierter Domänenname (FQDN) kann angegeben werden. Kann so eingestellt werden, dass IPv6-Adressen verwendet werden, wenn Trident mit dem IPv6-Flag installiert wurde. IPv6-Adressen müssen in eckigen Klammern angegeben werden, wie `[28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]`. Für einen reibungslosen MetroCluster-Switchover siehe MetroCluster Beispiel.	"10.0.0.1", "[2001:1234:abcd::fefe]"
`dataLIF`	IP-Adresse des Protokoll-LIF. NetApp empfiehlt, `dataLIF` anzugeben. Falls nicht angegeben, ruft Trident dataLIFs vom SVM ab. Ein vollqualifizierter Domainname (FQDN) kann für die NFS-Mount-Operationen angegeben werden, wodurch die Erstellung eines Round-Robin-DNS zur Lastverteilung über mehrere dataLIFs ermöglicht wird. Kann nach der Ersteinrichtung geändert werden. Siehe . Kann so eingestellt werden, dass IPv6-Adressen verwendet werden, wenn Trident mit dem IPv6-Flag installiert wurde. IPv6-Adressen müssen in eckigen Klammern angegeben werden, wie `[28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]`. **Für MetroCluster weglassen.** Siehe MetroCluster Beispiel.	Angegebene Adresse oder abgeleitet von SVM, falls nicht angegeben (nicht empfohlen)
`svm`	Zu verwendende virtuelle Speichermaschine **Für MetroCluster auslassen.** Siehe MetroCluster Beispiel.	Wird abgeleitet, wenn eine SVM `managementLIF` angegeben ist
`autoExportPolicy`	Automatische Erstellung und Aktualisierung von Exportrichtlinien aktivieren [Boolean]. Mit den Optionen `autoExportPolicy` und `autoExportCIDRs` kann Trident Exportrichtlinien automatisch verwalten.	false
`autoExportCIDRs`	Liste der CIDRs, anhand derer die Node-IPs von Kubernetes gefiltert werden, wenn `autoExportPolicy` aktiviert ist. Mit den Optionen `autoExportPolicy` und `autoExportCIDRs` kann Trident Exportrichtlinien automatisch verwalten.	["0.0.0.0/0", "::/0"]`
`labels`	Satz beliebiger JSON-formatierter Bezeichnungen, die auf Volumes angewendet werden sollen	""

Parameter	Beschreibung	Standard
`clientCertificate`	Base64-kodierter Wert des Clientzertifikats. Wird für zertifikatbasierte Authentifizierung verwendet	""
`clientPrivateKey`	Base64-kodierter Wert des privaten Client-Schlüssels. Wird für zertifikatbasierte Authentifizierung verwendet	""
`trustedCACertificate`	Base64-kodierter Wert des vertrauenswürdigen CA-Zertifikats. Optional. Wird für zertifikatbasierte Authentifizierung verwendet	""
`username`	Benutzername für die Verbindung zum Cluster/SVM. Wird für die anmeldeinformationsbasierte Authentifizierung verwendet. Für die Active Directory Authentifizierung siehe "Authentifizieren Sie Trident bei einer Backend-SVM mit Active Directory-Anmeldeinformationen".	
`password`	Passwort für die Verbindung zum Cluster/SVM. Wird für die anmeldeinformationsbasierte Authentifizierung verwendet. Für die Active Directory Authentifizierung siehe "Authentifizieren Sie Trident bei einer Backend-SVM mit Active Directory-Anmeldeinformationen".	
`storagePrefix`	Präfix, das beim Bereitstellen neuer Volumes in der SVM verwendet wird. Kann nach der Festlegung nicht mehr geändert werden. **HINWEIS** Bei Verwendung von ontap-nas-economy und einem storagePrefix, der 24 oder mehr Zeichen umfasst, wird das Speicherpräfix nicht in die Qtrees eingebettet, obwohl es im Volume-Namen enthalten ist.	"Trident"

Parameter	Beschreibung	Standard
`aggregate`	Aggregat für die Bereitstellung (optional; falls festgelegt, muss es der SVM zugewiesen werden). Für den `ontap-nas-flexgroup` Treiber wird diese Option ignoriert. Wenn nicht zugewiesen, kann eines der verfügbaren Aggregate zur Bereitstellung eines FlexGroup Volumes verwendet werden. **HINWEIS** Wird das Aggregat in SVM aktualisiert, wird es in Trident automatisch durch Abfragen von SVM aktualisiert, ohne dass der Trident Controller neu gestartet werden muss. Wenn Sie in Trident ein bestimmtes Aggregat für die Bereitstellung von Volumes konfiguriert haben und dieses Aggregat umbenannt oder aus der SVM verschoben wird, wechselt das Backend in Trident beim Abfragen des SVM-Aggregats in den Fehlerzustand. Sie müssen entweder das Aggregat auf eines ändern, das auf der SVM vorhanden ist, oder es vollständig entfernen, um das Backend wieder online zu bringen. **Nicht für AFX storage systems angeben.**	""
`limitAggregateUsage`	Die Bereitstellung schlägt fehl, wenn die Nutzung diesen Prozentsatz überschreitet. **Gilt nicht für Amazon FSx for ONTAP**. **Nicht für AFX storage systems angeben.**	"" (wird nicht standardmäßig erzwungen)

Parameter	Beschreibung	Standard
flexgroupAggregateList	Liste der Aggregate für die Bereitstellung (optional; falls festgelegt, muss sie der SVM zugewiesen werden). Alle Aggregate, die der SVM zugewiesen sind, werden zur Bereitstellung eines FlexGroup-Volumes verwendet. Unterstützt für den **ontap-nas-flexgroup**-Speichertreiber. **HINWEIS**: Wird die Aggregatliste in SVM aktualisiert, wird sie in Trident automatisch durch Abfrage von SVM aktualisiert, ohne dass der Trident Controller neu gestartet werden muss. Wenn Sie in Trident eine bestimmte Aggregatliste für die Volume-Bereitstellung konfiguriert haben und diese Aggregatliste umbenannt oder aus SVM verschoben wird, wechselt das Backend in Trident beim Abfragen der SVM-Aggregatliste in den Fehlerzustand. Sie müssen entweder die Aggregatliste auf eine ändern, die in SVM vorhanden ist, oder sie vollständig entfernen, um das Backend wieder online zu bringen.	""
`limitVolumeSize`	Die Bereitstellung schlägt fehl, wenn die angeforderte Volume-Größe diesen Wert überschreitet.	"" (wird nicht standardmäßig erzwungen)
`debugTraceFlags`	Debug-Flags zur Verwendung bei der Fehlersuche. Beispiel, {"api":false, "method":true} Verwenden Sie `debugTraceFlags` nicht, es sei denn, Sie führen eine Fehlersuche durch und benötigen eine detaillierte Protokollausgabe.	null
`nasType`	Konfigurieren Sie die Erstellung von NFS- oder SMB-Volumes. Optionen sind `nfs`, `smb` oder null. Die Einstellung auf null verwendet standardmäßig NFS-Volumes. **Falls angegeben, immer auf `nfs` für AFX-Speichersysteme setzen.**	`nfs`

Parameter	Beschreibung	Standard
`nfsMountOptions`	Kommagetrennte Liste von NFS-Mountoptionen. Die Mountoptionen für persistente Kubernetes-Volumes werden normalerweise in Speicherklassen angegeben, aber wenn keine Mountoptionen in einer Speicherklasse angegeben sind, verwendet Trident die Mountoptionen, die in der Konfigurationsdatei des Storage-Backends angegeben sind. Wenn weder in der Speicherklasse noch in der Konfigurationsdatei Mountoptionen angegeben sind, setzt Trident keine Mountoptionen für das zugehörige persistente Volume.	""
`qtreesPerFlexvol`	Maximale Qtrees pro FlexVol, muss im Bereich [50, 300] liegen	"200"
`smbShare`	Sie können eines der folgenden angeben: den Namen einer SMB-Freigabe, die mit der Microsoft Management Console oder ONTAP CLI erstellt wurde; einen Namen, damit Trident die SMB-Freigabe erstellen kann; oder Sie können den Parameter leer lassen, um den gemeinsamen Freigabezugriff auf Volumes zu verhindern. Dieser Parameter ist optional für lokale ONTAP. Dieser Parameter ist für Amazon FSx for ONTAP Backends erforderlich und darf nicht leer sein.	`smb-share`
`useREST`	Boolescher Parameter zur Verwendung von ONTAP REST APIs. `useREST` Wenn auf `true` gesetzt, verwendet Trident ONTAP REST APIs zur Kommunikation mit dem Backend; wenn auf `false` gesetzt, verwendet Trident ONTAPI (ZAPI)-Aufrufe zur Kommunikation mit dem Backend. Diese Funktion erfordert ONTAP 9.11.1 und höher. Zusätzlich muss die verwendete ONTAP-Anmelderolle Zugriff auf die `ontapi` Anwendung haben. Dies wird durch die vordefinierten `vsadmin` und `cluster-admin` Rollen erfüllt. Ab der Trident 24.06-Version und ONTAP 9.15.1 oder höher ist `useREST` standardmäßig auf `true` gesetzt; ändern Sie `useREST` auf `false`, um ONTAPI (ZAPI)-Aufrufe zu verwenden. **Falls angegeben, immer auf `true` für AFX-Speichersysteme setzen.**	`true` für ONTAP 9.15.1 oder höher, andernfalls `false`.
`limitVolumePoolSize`	Maximal anforderbare FlexVol-Größe bei Verwendung von Qtrees im ontap-nas-economy Backend.	"" (wird nicht standardmäßig erzwungen)
`denyNewVolumePools`	Beschränkt `ontap-nas-economy` Backends darauf, neue FlexVol Volumes zu erstellen, um ihre Qtrees zu enthalten. Nur bereits vorhandene Flexvols werden für die Bereitstellung neuer PVs verwendet.	

Parameter	Beschreibung	Standard
`adAdminUser`	Active Directory-Administratorbenutzer oder -Benutzergruppe mit vollem Zugriff auf SMB-Freigaben. Verwenden Sie diesen Parameter, um Administratorrechte für die SMB-Freigabe mit vollständiger Kontrolle zu erteilen.	

Backend-Konfigurationsoptionen für die Bereitstellung von Volumes

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

Parameter	Beschreibung	Standard
`spaceAllocation`	Speicherplatzzuweisung für Qtrees	"true"
`spaceReserve`	Platzreservierungsmodus; "none" (dünn) oder "volume" (dick)	"none"
`snapshotPolicy`	Zu verwendende Snapshot-Richtlinie	"none"
`qosPolicy`	QoS-Richtliniengruppe, die für erstellte Volumes zugewiesen werden soll. Wählen Sie eine der qosPolicy oder adaptiveQosPolicy pro Speicherpool/Backend aus.	""
`adaptiveQosPolicy`	Adaptive QoS-Richtliniengruppe zur Zuweisung für erstellte Volumes. Wählen Sie eine von qosPolicy oder adaptiveQosPolicy pro Storage-Pool/Backend. Wird von ontap-nas-economy nicht unterstützt.	""
`snapshotReserve`	Prozentsatz des für Snapshots reservierten Volumens	"0", falls `snapshotPolicy`"none", andernfalls ""
`splitOnClone`	Trennen Sie einen Klon bei seiner Erstellung von seinem übergeordneten Objekt	"false"
`encryption`	Aktivieren Sie NetApp Volume Encryption (NVE) auf dem neuen Volume; Standard ist `false`. NVE muss auf dem Cluster lizenziert und aktiviert sein, um diese Option zu verwenden. Wenn NAE im Backend aktiviert ist, wird jedes in Trident bereitgestellte Volume NAE-aktiviert. Weitere Informationen finden Sie unter: "Wie Trident mit NVE und NAE zusammenarbeitet".	"false"
`tieringPolicy`	Tiering-Richtlinie auf "none" setzen	
`unixPermissions`	Modus für neue Volumes	"777" für NFS-Volumes; leer (nicht zutreffend) für SMB-Volumes
`snapshotDir`	Steuert den Zugriff auf das `.snapshot` Verzeichnis	`true`, `false` (explizit festlegen).
`exportPolicy`	Zu verwendende Exportrichtlinie	"default"

Parameter	Beschreibung	Standard
`securityStyle`	Sicherheitsstil für neue Volumes. NFS unterstützt `mixed` und `unix` Sicherheitsstile. SMB unterstützt `mixed` und `ntfs` Sicherheitsstile.	NFS-Standard ist `unix`. SMB-Standard ist `ntfs`.
`nameTemplate`	Vorlage zum Erstellen benutzerdefinierter Volume-Namen.	""

HINWEIS Die Verwendung von QoS-Richtliniengruppen mit Trident erfordert ONTAP 9.8 oder höher. Sie sollten eine nicht gemeinsam genutzte QoS-Richtliniengruppe verwenden und sicherstellen, dass die Richtliniengruppe auf jede Komponente einzeln angewendet wird. Eine gemeinsam genutzte QoS-Richtliniengruppe erzwingt die Obergrenze für den Gesamtdurchsatz aller Workloads.

Beispiele für die Volume-Bereitstellung

Hier ist 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` verwendet Trident nun eine neue Berechnung, um sicherzustellen, dass die FlexVol mit dem snapshotReserve-Prozentsatz und dem PVC korrekt dimensioniert ist. Wenn der Benutzer einen PVC anfordert, erstellt Trident das ursprüngliche FlexVol mit mehr Speicherplatz mithilfe der neuen Berechnung. Diese Berechnung stellt sicher, dass der Benutzer den beschreibbaren

Speicherplatz erhält, den er im PVC angefordert hat, und nicht weniger als angefordert. Vor v21.07, wenn der Benutzer einen PVC anfordert (zum Beispiel 5 GiB) und der snapshotReserve auf 50 Prozent gesetzt ist, erhält er nur 2,5 GiB beschreibbaren Speicherplatz. Dies liegt daran, dass das, was der Benutzer anfordert, das gesamte Volume ist und `snapshotReserve` ein Prozentsatz davon ist. Mit Trident 21.07 ist das, was der Benutzer anfordert, der beschreibbare Speicherplatz, und Trident definiert die `snapshotReserve` Zahl als Prozentsatz des gesamten Volumens. Dies gilt nicht für `ontap-nas-economy`. Siehe das folgende Beispiel, um zu sehen, wie dies funktioniert:

Die Berechnung erfolgt wie folgt:

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

Für snapshotReserve = 50 %, und eine PVC-Anforderung = 5 GiB, beträgt die Gesamtgröße des Volumes 5/0,5 = 10 GiB und die verfügbare Größe 5 GiB, was der vom Benutzer in der PVC-Anforderung angeforderten Größe entspricht. Der `volume show`-Befehl sollte Ergebnisse ähnlich diesem Beispiel anzeigen:

Vorhandene Backends aus früheren Installationen werden beim Upgrade von Trident Volumes wie oben beschrieben bereitstellen. Für Volumes, die Sie vor dem Upgrade erstellt haben, sollten Sie deren Größe anpassen, damit die Änderung wirksam wird. Beispielsweise führte ein 2-GiB-PVC mit `snapshotReserve=50` zuvor zu einem Volume, das 1 GiB beschreibbaren Speicherplatz bereitstellt. Durch die Größenänderung des Volumes auf beispielsweise 3 GiB erhält die Anwendung 3 GiB beschreibbaren Speicherplatz auf einem 6-GiB-Volume.

Minimale Konfigurationsbeispiele

Die folgenden Beispiele zeigen Basiskonfigurationen, bei denen die meisten Parameter auf Standardwerte eingestellt sind. Dies ist die einfachste Methode, ein Backend zu definieren.

HINWEIS	Wenn Sie Amazon FSx auf NetApp ONTAP mit Trident verwenden, wird empfohlen, für LIFs DNS-Namen anstelle von IP-Adressen anzugeben.

ONTAP NAS economy Beispiel

```
---
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 FlexGroup-Beispiel

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

MetroCluster Beispiel

Sie können das Backend so konfigurieren, dass Sie die Backend-Definition nach einem Switchover und Switchback während "SVM-Replikation und -Wiederherstellung" nicht manuell aktualisieren müssen.

Für einen nahtlosen Switchover und Switchback geben Sie die SVM mit `managementLIF` an und lassen Sie die `dataLIF` und `svm` Parameter weg. Beispiel:

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

Beispiel für SMB-Volumes

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

Beispiel für zertifikatsbasierte Authentifizierung

Dies ist ein minimales Beispiel für eine Backend-Konfiguration. `clientCertificate`, `clientPrivateKey` und `trustedCACertificate` (optional, falls eine vertrauenswürdige Zertifizierungsstelle verwendet wird) werden in `backend.json` befüllt und nehmen die Base64-kodierten Werte des Clientzertifikats, des privaten Schlüssels und des Zertifikats der vertrauenswürdigen Zertifizierungsstelle an.

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

Beispiel für eine automatische Exportrichtlinie

Dieses Beispiel zeigt Ihnen, wie Sie Trident anweisen können, dynamische Exportrichtlinien zu verwenden, um die Exportrichtlinie automatisch zu erstellen und zu verwalten. Dies funktioniert gleichermaßen für die `ontap-nas-economy` und `ontap-nas-flexgroup` Treiber.

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

Beispiel für IPv6 addresses

Dieses Beispiel zeigt `managementLIF` die Verwendung einer IPv6-Adresse.

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

Amazon FSx für ONTAP mit SMB-Volumes Beispiel

Der `smbShare` Parameter ist für FSx for ONTAP mit SMB-Volumes erforderlich.

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

Beispiel für die Backend-Konfiguration mit nameTemplate

```
---
version: 1
storageDriverName: ontap-nas
backendName: ontap-nas-backend
managementLIF: <ip address>
svm: svm0
username: <admin>
password: <password>
defaults:
  nameTemplate:
"{{.volume.Name}}_{{.labels.cluster}}_{{.volume.Namespace}}_{{.vo\
    lume.RequestName}}"
labels:
  cluster: ClusterA
  PVC: "{{.volume.Namespace}}_{{.volume.RequestName}}"
```

Beispiele für Backends mit virtuellen Pools

In den unten gezeigten Beispiel-Backend-Definitionsdateien sind für alle Speicherpools spezifische Standardwerte festgelegt, wie `spaceReserve` bei none, `spaceAllocation` bei false und `encryption` bei false. Die virtuellen Pools werden im Speicherabschnitt definiert.

Trident legt Bereitstellungsbezeichnungen im Feld „Kommentare“ fest. Kommentare werden auf FlexVol für `ontap-nas` oder auf FlexGroup für `ontap-nas-flexgroup` festgelegt. Trident kopiert alle auf einem

virtuellen Pool vorhandenen Bezeichnungen bei der Bereitstellung auf das Speichervolume. Zur Vereinfachung können Speicheradministratoren Bezeichnungen pro virtuellem Pool definieren und Volumes nach Bezeichnung gruppieren.

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

ONTAP NAS-Beispiel

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

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

ONTAP NAS FlexGroup Beispiel

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

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

ONTAP NAS economy Beispiel

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

```
      encryption: "false"
      unixPermissions: "0775"
```

Backends zu StorageClasses zuordnen

Die folgenden StorageClass-Definitionen beziehen sich auf Beispiele für Backends mit virtuellen Pools. Mithilfe des `parameters.selector`-Feldes gibt jede StorageClass an, welche virtuellen Pools zum Hosten eines Volumes verwendet werden können. Das Volume weist die im gewählten virtuellen Pool definierten Aspekte auf.

- Die `protection-gold` StorageClass wird dem ersten und zweiten virtuellen Pool im `ontap-nas-flexgroup` Backend zugeordnet. Diese sind die einzigen Pools, die Schutz auf Goldniveau bieten.

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

- Die `protection-not-gold` StorageClass wird dem dritten und vierten virtuellen Pool im `ontap-nas-flexgroup` Backend zugeordnet. Dies sind die einzigen Pools, die ein anderes Schutzniveau als Gold bieten.

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

- Die `app-mysqldb` StorageClass wird dem vierten virtuellen Pool im `ontap-nas` Backend zugeordnet. Dies ist der einzige Pool, der eine Speicherpoolkonfiguration für mysqldb Typ App bietet.

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

- Der `protection-silver-creditpoints-20k` StorageClass wird dem dritten virtuellen Pool im `ontap-nas-flexgroup` Backend zugeordnet. Dies ist der einzige Pool, der Schutz auf Silber-Niveau und 20000 Kreditpunkte bietet.

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

- Die `creditpoints-5k` StorageClass wird dem dritten virtuellen Pool im `ontap-nas` Backend und dem zweiten virtuellen Pool im `ontap-nas-economy` Backend zugeordnet. Dies sind die einzigen Poolangebote mit 5000 Kreditpunkten.

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

Trident entscheidet, welcher virtuelle Pool ausgewählt wird und stellt sicher, dass die Speicheranforderung erfüllt wird.

Aktualisieren `dataLIF` nach der Erstkonfiguration

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

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

HINWEIS	Wenn PVCs an einem oder mehreren Pods angeschlossen sind, müssen Sie alle entsprechenden Pods herunterfahren und anschließend wieder hochfahren, damit die neue dataLIF wirksam wird.

Beispiele für sicheres SMB

Backend-Konfiguration mit dem ontap-nas-Treiber

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-nas
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-nas
  managementLIF: 10.0.0.1
  svm: svm2
  nasType: smb
  defaults:
    adAdminUser: tridentADtest
  credentials:
    name: backend-tbc-ontap-invest-secret
```

Backend-Konfiguration mit ontap-nas-economy-Treiber

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-nas
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-nas-economy
  managementLIF: 10.0.0.1
  svm: svm2
  nasType: smb
  defaults:
    adAdminUser: tridentADtest
  credentials:
    name: backend-tbc-ontap-invest-secret
```

Backend-Konfiguration mit Speicherpool

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-nas
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-nas
  managementLIF: 10.0.0.1
  svm: svm0
  useREST: false
  storage:
  - labels:
      app: msoffice
    defaults:
      adAdminUser: tridentADuser
  nasType: smb
  credentials:
    name: backend-tbc-ontap-invest-secret
```

Speicherklassenbeispiel mit dem ontap-nas-Treiber

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-smb-sc
  annotations:
    trident.netapp.io/smbShareAdUserPermission: change
    trident.netapp.io/smbShareAdUser: tridentADtest
parameters:
  backendType: ontap-nas
  csi.storage.k8s.io/node-stage-secret-name: smbcreds
  csi.storage.k8s.io/node-stage-secret-namespace: trident
  trident.netapp.io/nasType: smb
provisioner: csi.trident.netapp.io
reclaimPolicy: Delete
volumeBindingMode: Immediate
```

HINWEIS	Stellen Sie sicher, dass Sie `annotations` hinzufügen, um Secure SMB zu aktivieren. Secure SMB funktioniert ohne die Annotationen nicht, unabhängig von den im Backend oder PVC festgelegten Konfigurationen.

Speicherklassenbeispiel mit ontap-nas-economy-Treiber

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-smb-sc
  annotations:
    trident.netapp.io/smbShareAdUserPermission: change
    trident.netapp.io/smbShareAdUser: tridentADuser3
parameters:
  backendType: ontap-nas-economy
  csi.storage.k8s.io/node-stage-secret-name: smbcreds
  csi.storage.k8s.io/node-stage-secret-namespace: trident
  trident.netapp.io/nasType: smb
provisioner: csi.trident.netapp.io
reclaimPolicy: Delete
volumeBindingMode: Immediate
```

PVC-Beispiel mit einem einzelnen AD-Benutzer

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-pvc4
  namespace: trident
  annotations:
    trident.netapp.io/smbShareAccessControl: |
      change:
        - tridentADtest
      read:
        - tridentADuser
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: ontap-smb-sc
```

PVC-Beispiel mit mehreren AD-Benutzern

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-test-pvc
  annotations:
    trident.netapp.io/smbShareAccessControl: |
      full_control:
        - tridentTestuser
        - tridentuser
        - tridentTestuser1
        - tridentuser1
      change:
        - tridentADuser
        - tridentADuser1
        - tridentADuser4
        - tridentTestuser2
      read:
        - tridentTestuser2
        - tridentTestuser3
        - tridentADuser2
        - tridentADuser3
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
```

Amazon FSx for NetApp ONTAP

Verwenden Sie Trident mit Amazon FSx for NetApp ONTAP

"Amazon FSx for NetApp ONTAP" ist ein vollständig verwalteter AWS-Service, der Dateisysteme betreibt, die vom NetApp ONTAP-Speicherbetriebssystem unterstützt werden. Er bietet ONTAP-Funktionen, Leistung und Administration mit der Skalierbarkeit und dem einfachen Betrieb von AWS. Ein Dateisystem ist die primäre Ressource in Amazon FSx und entspricht einem lokalen ONTAP-Cluster. Jedes Dateisystem enthält eine oder mehrere Storage Virtual Machines (SVMs), und jede SVM enthält ein oder mehrere Volumes, die Dateien und Verzeichnisse speichern. Diese Integration ermöglicht es Kubernetes-Clustern, die in Amazon Elastic Kubernetes Service (EKS) ausgeführt werden, ONTAP-gestützte persistente Volumes für Block- und Dateiworkloads bereitzustellen.

Anforderungen

Zusätzlich zu "Trident-Anforderungen" benötigen Sie zur Integration von FSx for ONTAP mit Trident Folgendes:

- Ein bestehender Amazon EKS-Cluster oder ein selbstverwalteter Kubernetes-Cluster mit `kubectl` installiert.
- Ein vorhandenes Amazon FSx for NetApp ONTAP Dateisystem und eine Storage Virtual Machine (SVM), die von den Worker-Knoten Ihres Clusters aus erreichbar ist.
- Worker-Knoten, die für "NFS oder iSCSI" vorbereitet sind.

HINWEIS	Stellen Sie sicher, dass Sie die für Amazon Linux und Ubuntu "Amazon Machine Images" (AMIs) erforderlichen Node-Vorbereitungsschritte je nach Ihrem EKS-AMI-Typ befolgen.

Überlegungen

- SMB-Volumes:
  - SMB-Volumes werden nur mit dem `ontap-nas` Treiber unterstützt.
  - SMB-Volumes werden mit dem Trident EKS-Add-on nicht unterstützt.
  - Trident unterstützt SMB-Volumes nur, wenn sie in Pods eingebunden sind, die auf Windows-Knoten ausgeführt werden. Siehe "Bereiten Sie die Bereitstellung von SMB-Volumes vor" für Details.
- Vor Trident 24.02 konnten Volumes, die auf Amazon FSx-Dateisystemen mit aktivierten automatischen Backups erstellt wurden, nicht von Trident gelöscht werden. Um dieses Problem in Trident 24.02 oder höher zu verhindern, geben Sie die `fsxFilesystemID`, AWS `apiRegion`, AWS `apikey` und AWS `secretKey` in der Backend-Konfigurationsdatei für AWS FSx for ONTAP an.

HINWEIS	Wenn Sie eine IAM-Rolle für Trident angeben, können Sie das explizite Angeben der Felder `apiRegion`, `apiKey` und `secretKey` für Trident weglassen. Weitere Informationen finden sich unter "FSx for ONTAP Konfigurationsoptionen und Beispiele".

Gleichzeitige Verwendung von Trident SAN/iSCSI und EBS-CSI-Treiber

Wenn Sie ontap-san-Treiber (z. B. iSCSI) mit AWS (EKS, ROSA, EC2 oder einer anderen Instanz) verwenden möchten, kann die erforderliche Multipath-Konfiguration auf den Knoten mit dem Amazon Elastic Block Store (EBS) CSI-Treiber in Konflikt geraten. Um sicherzustellen, dass Multipathing ohne Beeinträchtigung der EBS-Festplatten auf demselben Knoten funktioniert, müssen Sie EBS in Ihrer Multipathing-Konfiguration ausschließen. Dieses Beispiel zeigt eine `multipath.conf` Datei, die die erforderlichen Trident-Einstellungen enthält und gleichzeitig EBS-Festplatten vom Multipathing ausschließt:

```
defaults {
    find_multipaths no
}
blacklist {
    device {
        vendor "NVME"
        product "Amazon Elastic Block Store"
    }
}
```

Authentifizierung

Trident bietet zwei Modi der Authentifizierung.

- Anmeldeinformationsbasiert (Empfohlen): Speichert Anmeldeinformationen sicher im AWS Secrets Manager. Sie können den `fsxadmin` Benutzer für Ihr Dateisystem oder den `vsadmin` Benutzer, der für Ihre SVM konfiguriert ist, verwenden.

WARNUNG	Trident erwartet, als `vsadmin` SVM-Benutzer oder als Benutzer mit einem anderen Namen, aber derselben Rolle ausgeführt zu werden. Amazon FSx for NetApp ONTAP hat einen `fsxadmin` Benutzer, der ein eingeschränkter Ersatz für den ONTAP `admin` Cluster-Benutzer ist. Wir empfehlen dringend, `vsadmin` mit Trident zu verwenden.

- Zertifikatsbasiert: Trident wird mit der SVM auf Ihrem FSx-Dateisystem unter Verwendung eines auf Ihrer SVM installierten Zertifikats kommunizieren.

Einzelheiten zur Aktivierung der Authentifizierung finden Sie in der Authentifizierung für Ihren Treibertyp:

- "ONTAP NAS Authentifizierung"
- "ONTAP SAN Authentifizierung"

Getestete Amazon Machine Images (AMIs)

Der EKS-Cluster unterstützt verschiedene Betriebssysteme, aber AWS hat bestimmte Amazon Machine Images (AMIs) für Container und EKS optimiert. Die folgenden AMIs wurden mit NetApp Trident 25.02 getestet.

AMI	NAS	NAS-economy	iSCSI	iSCSI-economy
AL2023_x86_64_STANDARD	Ja	Ja	Ja	Ja
AL2_x86_64	Ja	Ja	Ja*	Ja*
BOTTLEROCKET_x86_64	Ja**	Ja	k. A.	k. A.
AL2023_ARM_64_STANDARD	Ja	Ja	Ja	Ja
AL2_ARM_64	Ja	Ja	Ja*	Ja*

BOTTLEROCKET_ARM_64	Ja**	Ja	k. A.	k. A.

- * Das PV kann nicht ohne Neustart des Knotens gelöscht werden
- ** Funktioniert nicht mit NFSv3 mit Trident Version 25.02.

HINWEIS	Wenn Ihr gewünschtes AMI hier nicht aufgeführt ist, bedeutet das nicht, dass es nicht unterstützt wird; es wurde lediglich noch nicht getestet. Diese Liste dient als Orientierungshilfe für AMIs, von denen bekannt ist, dass sie funktionieren.

Tests durchgeführt mit:

- EKS-Version: 1.32
- Installationsmethode: Helm 25.06 und als AWS add-On 25.06
- Für NAS wurden sowohl NFSv3 als auch NFSv4.1 getestet.
- Für SAN wurde nur iSCSI getestet, nicht NVMe-oF.

Durchgeführte Tests:

- Erstellen: Speicherklasse, pvc, pod
- Löschen: pod, pvc (regulär, qtree/lun – economy, NAS mit AWS backup)

Weitere Informationen

- "Amazon FSx for NetApp ONTAP Dokumentation"
- "Blogpost zu Amazon FSx for NetApp ONTAP"

Erstellen Sie eine IAM-Rolle und ein AWS-Geheimnis

Sie können Kubernetes-Pods so konfigurieren, dass sie auf AWS-Ressourcen zugreifen, indem sie sich als AWS IAM-Rolle authentifizieren, anstatt explizite AWS-Anmeldeinformationen anzugeben.

HINWEIS	Zur Authentifizierung mit einer AWS IAM-Rolle müssen Sie einen Kubernetes-Cluster haben, der mit EKS bereitgestellt wurde.

AWS Secrets Manager-Geheimnis erstellen

Da Trident APIs gegen einen FSx vserver ausführt, um den Speicher für Sie zu verwalten, benötigt es hierfür Anmeldeinformationen. Der sichere Weg, diese Anmeldeinformationen zu übermitteln, ist über ein AWS Secrets Manager secret. Wenn Sie also noch keines haben, müssen Sie ein AWS Secrets Manager secret erstellen, das die Anmeldeinformationen für das vsadmin account enthält.

Dieses Beispiel erstellt ein AWS Secrets Manager secret zum Speichern von Trident CSI-Anmeldeinformationen:

```
aws secretsmanager create-secret --name trident-secret --description
"Trident CSI credentials"\
  --secret-string
"{\"username\":\"vsadmin\",\"password\":\"<svmpassword>\"}"
```

IAM-Richtlinie erstellen

Trident benötigt außerdem AWS-Berechtigungen, um korrekt zu funktionieren. Daher müssen Sie eine Richtlinie erstellen, die Trident die erforderlichen Berechtigungen erteilt.

Die folgenden Beispiele erstellen eine IAM-Richtlinie mithilfe der AWS CLI:

```
aws iam create-policy --policy-name AmazonFSxNCSIDriverPolicy --policy
-document file://policy.json
  --description "This policy grants access to Trident CSI to FSxN and
Secrets manager"
```

Policy JSON-Beispiel:

```
{
  "Statement": [
    {
      "Action": [
        "fsx:DescribeFileSystems",
        "fsx:DescribeVolumes",
        "fsx:CreateVolume",
        "fsx:RestoreVolumeFromSnapshot",
        "fsx:DescribeStorageVirtualMachines",
        "fsx:UntagResource",
        "fsx:UpdateVolume",
        "fsx:TagResource",
        "fsx:DeleteVolume"
      ],
      "Effect": "Allow",
      "Resource": "*"
    },
    {
      "Action": "secretsmanager:GetSecretValue",
      "Effect": "Allow",
      "Resource": "arn:aws:secretsmanager:<aws-region>:<aws-account-
id>:secret:<aws-secret-manager-name>*"
    }
  ],
  "Version": "2012-10-17"
}
```

Pod-Identität oder IAM-Rolle für die Service account association (IRSA) erstellen

Sie können ein Kubernetes-Servicekonto so konfigurieren, dass es eine AWS Identity and Access Management (IAM)-Rolle mit EKS Pod Identity oder IAM role for Service account association (IRSA) übernimmt. Alle Pods, die für die Verwendung des Servicekontos konfiguriert sind, können dann auf alle AWS-Services zugreifen, auf die die Rolle Berechtigungen hat.

Pod-Identität

Amazon EKS Pod Identity associations bieten die Möglichkeit, Anmeldeinformationen für Ihre Anwendungen zu verwalten, ähnlich wie Amazon EC2 instance profiles Anmeldeinformationen für Amazon EC2 instances bereitstellen.

Installieren Sie Pod Identity auf Ihrem EKS-Cluster:

Sie können eine Pod-Identität über die AWS-Konsole oder mit dem folgenden AWS CLI-Befehl erstellen:

```
aws eks create-addon --cluster-name <EKS_CLUSTER_NAME> --addon-name
eks-pod-identity-agent
```

Weitere Informationen finden Sie unter "Richten Sie den Amazon EKS Pod Identity Agent ein".

Erstellen Sie trust-relationship.json:

Erstellen Sie trust-relationship.json, um dem EKS Service Principal zu ermöglichen, diese Rolle für Pod Identity zu übernehmen. Erstellen Sie dann eine Rolle mit dieser Vertrauensrichtlinie:

```
aws iam create-role \
  --role-name fsxn-csi-role --assume-role-policy-document file://trust-
relationship.json \
  --description "fsxn csi pod identity role"
```

trust-relationship.json-Datei:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "pods.eks.amazonaws.com"
      },
      "Action": [
        "sts:AssumeRole",
        "sts:TagSession"
      ]
    }
  ]
}
```

Die Rollenrichtlinie an die IAM-Rolle anhängen:

Hängen Sie die Rollenrichtlinie aus dem vorherigen Schritt an die erstellte IAM-Rolle an:

```
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:111122223333:policy/fsxn-csi-policy \
  --role-name fsxn-csi-role
```

Eine Pod-Identitätszuordnung erstellen:

Erstellen Sie eine Pod-Identitätszuordnung zwischen der IAM-Rolle und dem Trident-Dienstkonto (trident-controller)

```
aws eks create-pod-identity-association \
  --cluster-name <EKS_CLUSTER_NAME> \
  --role-arn arn:aws:iam::111122223333:role/fsxn-csi-role \
  --namespace trident --service-account trident-controller
```

IAM role für Service account association (IRSA)

Verwendung der AWS CLI:

```
aws iam create-role --role-name AmazonEKS_FSxN_CSI_DriverRole \
  --assume-role-policy-document file://trust-relationship.json
```

trust-relationship.json-Datei:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws:iam::<account_id>:oidc-
provider/<oidc_provider>"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringEquals": {
          "<oidc_provider>:aud": "sts.amazonaws.com",
          "<oidc_provider>:sub":
"system:serviceaccount:trident:trident-controller"
        }
      }
    }
  ]
}
```

Aktualisieren Sie die folgenden Werte in der `trust-relationship.json` Datei:

- **<account_id>** - Ihre AWS-Konto-ID
- **<oidc_provider>** - Die OIDC Ihres EKS-Clusters. Sie können den oidc_provider durch Ausführen des folgenden Befehls abrufen:

```
aws eks describe-cluster --name my-cluster --query
"cluster.identity.oidc.issuer"\
  --output text | sed -e "s/^https:\/\///"
```

Verknüpfen Sie die IAM-Rolle mit der IAM-Richtlinie:

Sobald die Rolle erstellt wurde, hängen Sie die Richtlinie (die im vorherigen Schritt erstellt wurde) mit diesem Befehl an die Rolle an:

```
aws iam attach-role-policy --role-name my-role --policy-arn <IAM policy
ARN>
```

Überprüfen Sie, ob der OICD provider zugeordnet ist:

Überprüfen Sie, ob Ihr OIDC-Anbieter mit Ihrem Cluster verknüpft ist. Sie können dies mit diesem Befehl überprüfen:

```
aws iam list-open-id-connect-providers | grep $oidc_id | cut -d "/" -f4
```

Wenn die Ausgabe leer ist, verwenden Sie den folgenden Befehl, um IAM OIDC mit Ihrem Cluster zu verknüpfen:

```
eksctl utils associate-iam-oidc-provider --cluster $cluster_name
--approve
```

Wenn Sie eksctl verwenden, verwenden Sie das folgende Beispiel, um eine IAM-Rolle für das Dienstkonto in EKS zu erstellen:

```
eksctl create iamserviceaccount --name trident-controller --namespace
trident \
  --cluster <my-cluster> --role-name AmazonEKS_FSxN_CSI_DriverRole
--role-only \
  --attach-policy-arn <IAM-Policy ARN> --approve
```

Trident installieren

Trident optimiert Amazon FSx for NetApp ONTAP Speicherverwaltung in Kubernetes, damit sich Ihre Entwickler und Administratoren auf die Anwendungsbereitstellung konzentrieren können. Sie können Trident mit einer der folgenden Methoden installieren:

- Helm
- EKS Add-on

Wenn Sie die Snapshot-Funktionalität nutzen möchten, installieren Sie das CSI Snapshot Controller-Add-on. Weitere Informationen sind unter "Snapshot-Funktionalität für CSI-Volumes aktivieren" verfügbar.

Trident über helm installieren

Pod-Identität

1. Fügen Sie das Trident Helm repository hinzu:

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

2. Installieren Sie Trident anhand des folgenden Beispiels:

```
helm install trident-operator netapp-trident/trident-operator --version 100.2502.1 --namespace trident --create-namespace
```

Sie können den `helm list` Befehl verwenden, um Installationsdetails wie Name, Namespace, Chart, Status, App-Version und Revisionsnummer zu überprüfen.

```
helm list -n trident
```

```
NAME                NAMESPACE   REVISION    UPDATED
STATUS       CHART                           APP VERSION

trident-operator    trident     1           2024-10-14
14:31:22.463122 +0300 IDT    deployed     trident-operator-
100.2502.0    25.02.0
```

Service-Account-Zuordnung (IRSA)

1. Fügen Sie das Trident Helm repository hinzu:

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

2. Legen Sie die Werte für **cloud provider** und **cloud identity** fest:

```
helm install trident-operator netapp-trident/trident-operator
--version 100.2502.1  \
--set cloudProvider="AWS" \
--set cloudIdentity="'eks.amazonaws.com/role-arn:
arn:aws:iam::<accountID>:role/<AmazonEKS_FSxN_CSI_DriverRole>'" \
--namespace trident \
--create-namespace
```

Sie können den `helm list` Befehl verwenden, um Installationsdetails wie Name, Namespace, Chart, Status, App-Version und Revisionsnummer zu überprüfen.

```
helm list -n trident
```

```
NAME                NAMESPACE   REVISION    UPDATED
STATUS       CHART                           APP VERSION

trident-operator    trident     1           2024-10-14
14:31:22.463122 +0300 IDT    deployed     trident-operator-
100.2510.0    25.10.0
```

HINWEIS

Wenn Sie iSCSI verwenden möchten, stellen Sie sicher, dass iSCSI auf Ihrem Client-Rechner aktiviert ist. Wenn Sie das AL2023 Worker node OS verwenden, können Sie die Installation des iSCSI-Clients automatisieren, indem Sie den node prep Parameter in der helm Installation hinzufügen:

```
helm install trident-operator netapp-trident/trident-
operator --version 100.2502.1 --namespace trident --create
-namespace --set nodePrep={iscsi}
```

Installieren Sie Trident über das EKS-Add-on

Das Trident EKS-Add-on enthält die neuesten Sicherheitspatches, Fehlerbehebungen und ist von AWS für die Verwendung mit Amazon EKS validiert. Das EKS-Add-on ermöglicht es Ihnen, konsequent sicherzustellen, dass Ihre Amazon EKS-Cluster sicher und stabil sind, und reduziert den Arbeitsaufwand, den Sie für die Installation, Konfiguration und Aktualisierung von Add-ons aufwenden müssen.

Voraussetzungen

Stellen Sie sicher, dass Sie Folgendes haben, bevor Sie das Trident-Add-on für AWS EKS konfigurieren:

- Ein Amazon EKS-Clusterkonto mit Add-on-Abonnement
- AWS-Berechtigungen für den AWS Marketplace:
  `"aws-marketplace:ViewSubscriptions",`
  `"aws-marketplace:Subscribe",`
  `"aws-marketplace:Unsubscribe`
- AMI type: Amazon Linux 2 (AL2_x86_64) oder Amazon Linux 2 Arm (AL2_ARM_64)
- Knotentyp: AMD oder ARM
- Ein bestehendes Amazon FSx for NetApp ONTAP file system

Aktivieren Sie das Trident Add-on für AWS

Verwaltungskonsole

1. Öffnen Sie die Amazon EKS-Konsole unter https://console.aws.amazon.com/eks/home#/clusters.
2. Wählen Sie im linken Navigationsbereich **Clusters** aus.
3. Wählen Sie den Namen des Clusters aus, für den Sie das NetApp Trident CSI-Add-on konfigurieren möchten.
4. Wählen Sie **Add-ons** und dann **Weitere Add-ons abrufen**.
5. Führen Sie die folgenden Schritte aus, um das Add-on auszuwählen:
   a. Scrollen Sie nach unten zum Abschnitt **AWS Marketplace add-ons** und geben Sie **"Trident"** in das Suchfeld ein.
   b. Aktivieren Sie das Kästchen in der oberen rechten Ecke des Trident by NetApp-Feldes.
   c. Wählen Sie **Next**.
6. Gehen Sie auf der Seite **Ausgewählte Add-ons konfigurieren** wie folgt vor:

HINWEIS	**Überspringen Sie diese Schritte, wenn Sie die Pod Identity association verwenden.**

   a. Wählen Sie die **Version** aus, die Sie verwenden möchten.
   b. Wenn Sie die IRSA-Authentifizierung verwenden, stellen Sie sicher, dass Sie die in den Optionalen Konfigurationseinstellungen verfügbaren Konfigurationswerte festlegen:
      - Wählen Sie die **Version** aus, die Sie verwenden möchten.
      - Folgen Sie dem **Add-on-Konfigurationsschema** und legen Sie den **configurationValues** -Parameter im Abschnitt **Konfigurationswerte** auf den role-arn fest, den Sie im vorherigen Schritt erstellt haben (Wert sollte folgendes Format haben):

```
{

  "cloudIdentity": "'eks.amazonaws.com/role-arn: <role ARN>'",
  "cloudProvider": "AWS"

}
```

+
Wenn Sie bei der Konfliktlösungsmethode „Überschreiben“ auswählen, kann eine oder mehrere Einstellungen des bestehenden Add-ons mit den Amazon EKS add-on Einstellungen überschrieben werden. Wenn Sie diese Option nicht aktivieren und ein Konflikt mit Ihren bestehenden Einstellungen auftritt, schlägt der Vorgang fehl. Sie können die resultierende Fehlermeldung zur Fehlerbehebung des Konflikts verwenden. Bevor Sie diese Option auswählen, stellen Sie sicher, dass das Amazon EKS add-on keine Einstellungen verwaltet, die Sie selbst verwalten müssen.

7. Wählen Sie **Weiter**.
8. Auf der Seite **Review and add** wählen Sie **Create**.

   Nach Abschluss der Add-on-Installation sehen Sie Ihr installiertes Add-on.

AWS CLI

1. Erstellen Sie die `add-on.json` Datei:

Für Pod Identity verwenden Sie das folgende Format:

HINWEIS Verwenden Sie die

```
{
  "clusterName": "<eks-cluster>",
  "addonName": "netapp_trident-operator",
  "addonVersion": "v25.6.0-eksbuild.1",
}
```

Für die IRSA-Authentifizierung verwenden Sie das folgende Format:

```
{
  "clusterName": "<eks-cluster>",
  "addonName": "netapp_trident-operator",
  "addonVersion": "v25.6.0-eksbuild.1",
  "serviceAccountRoleArn": "<role ARN>",
  "configurationValues": {
    "cloudIdentity": "'eks.amazonaws.com/role-arn: <role ARN>'",
    "cloudProvider": "AWS"
  }
}
```

HINWEIS Ersetzen Sie `<role ARN>` durch den ARN der Rolle, die im vorherigen Schritt erstellt wurde.

2. Installieren Sie das Trident EKS-Add-on.

```
aws eks create-addon --cli-input-json file://add-on.json
```

eksctl

Der folgende Beispielbefehl installiert das Trident EKS add-on:

```
eksctl create addon --name netapp_trident-operator --cluster
<cluster_name> --force
```

Aktualisieren Sie das Trident EKS add-on

Verwaltungskonsole

1. Öffnen Sie die Amazon EKS-Konsole https://console.aws.amazon.com/eks/home#/clusters.
2. Wählen Sie im linken Navigationsbereich **Clusters** aus.
3. Wählen Sie den Namen des Clusters aus, für den Sie das NetApp Trident CSI Add-on aktualisieren möchten.
4. Wählen Sie die Registerkarte **Add-ons** aus.
5. Wählen Sie **Trident by NetApp** und dann **Bearbeiten** aus.
6. Gehen Sie auf der Seite **Trident konfigurieren von NetApp** wie folgt vor:
   a. Wählen Sie die **Version** aus, die Sie verwenden möchten.
   b. Erweitern Sie die **Optional configuration settings** und nehmen Sie bei Bedarf Anpassungen vor.
   c. Wählen Sie **Save changes**.

AWS CLI

Das folgende Beispiel aktualisiert das EKS add-on:

```
aws eks update-addon --cluster-name <eks_cluster_name> --addon-name
netapp_trident-operator --addon-version v25.6.0-eksbuild.1 \
  --service-account-role-arn <role-ARN> --resolve-conflict preserve \
  --configuration-values "{\"cloudIdentity\":
\"'eks.amazonaws.com/role-arn: <role ARN>'\"}"
```

eksctl

- Überprüfen Sie die aktuelle Version Ihres FSxN Trident CSI Add-on. Ersetzen Sie `my-cluster` durch Ihren Clusternamen.

  ```
  eksctl get addon --name netapp_trident-operator --cluster my-cluster
  ```

  Beispielausgabe:

```
NAME                          VERSION              STATUS    ISSUES
IAMROLE    UPDATE AVAILABLE    CONFIGURATION VALUES
netapp_trident-operator    v25.6.0-eksbuild.1    ACTIVE    0
{"cloudIdentity":"'eks.amazonaws.com/role-arn:
arn:aws:iam::139763910815:role/AmazonEKS_FSXN_CSI_DriverRole'"}
```

- Aktualisieren Sie das Add-on auf die Version, die unter UPDATE AVAILABLE in der Ausgabe des vorherigen Schritts zurückgegeben wurde.

  ```
  eksctl update addon --name netapp_trident-operator --version
  v25.6.0-eksbuild.1 --cluster my-cluster --force
  ```

Wenn Sie die `--force` Option entfernen und eine der Amazon EKS Add-on-Einstellungen mit Ihren bestehenden Einstellungen in Konflikt steht, schlägt die Aktualisierung des Amazon EKS Add-ons fehl; Sie erhalten eine Fehlermeldung, die Ihnen hilft, den Konflikt zu beheben. Bevor Sie diese Option festlegen, stellen Sie sicher, dass das Amazon EKS Add-on keine Einstellungen verwaltet, die Sie verwalten müssen, da diese Einstellungen mit dieser Option überschrieben werden. Weitere Informationen zu anderen Optionen für diese Einstellung finden Sie unter "Add-ons". Weitere Informationen zur Amazon EKS Kubernetes-Feldverwaltung finden Sie unter "Kubernetes-Feldmanagement".

Deinstallieren/entfernen Sie das Trident EKS Add-on

Sie haben zwei Optionen, ein Amazon EKS add-on zu entfernen:

- **Add-on-Software auf Ihrem Cluster beibehalten** – Diese Option entfernt die Amazon EKS-Verwaltung aller Einstellungen. Außerdem wird die Möglichkeit entfernt, dass Amazon EKS Sie über Updates benachrichtigt und das Amazon EKS-Add-on nach einer Aktualisierung automatisch aktualisiert. Die Add-on-Software selbst bleibt jedoch auf Ihrem Cluster erhalten. Mit dieser Option wird das Add-on zu einer selbstverwalteten Installation, anstatt ein Amazon EKS-Add-on zu sein. Mit dieser Option gibt es keine Ausfallzeiten für das Add-on. Behalten Sie die `--preserve` Option im Befehl bei, um das Add-on beizubehalten.
- **Entfernen Sie die Add-on-Software vollständig aus Ihrem Cluster** – NetApp empfiehlt, das Amazon EKS-Add-on nur dann aus Ihrem Cluster zu entfernen, wenn keine Ressourcen auf Ihrem Cluster davon abhängig sind. Entfernen Sie die `--preserve` Option aus dem `delete` Befehl, um das Add-on zu entfernen.

HINWEIS Wenn dem Add-on eine IAM-Konto zugeordnet ist, wird das IAM-Konto nicht entfernt.

Verwaltungskonsole

1. Öffnen Sie die Amazon EKS-Konsole unter https://console.aws.amazon.com/eks/home#/clusters.
2. Wählen Sie im linken Navigationsbereich **Clusters** aus.
3. Wählen Sie den Namen des Clusters aus, für den Sie das NetApp Trident CSI-Add-on entfernen möchten.
4. Wählen Sie die Registerkarte **Add-ons** und dann **Trident by NetApp**.*
5. Wählen Sie **Remove**.
6. Führen Sie im Dialogfeld **Remove netapp_trident-operator confirmation** die folgenden Schritte aus:
   a. Wenn Amazon EKS die Einstellungen für das Add-on nicht mehr verwalten soll, wählen Sie **Preserve on cluster**. Tun Sie dies, wenn Sie die Add-on-Software auf Ihrem Cluster behalten und alle Einstellungen des Add-ons selbst verwalten möchten.
   b. Geben Sie **netapp_trident-operator** ein.
   c. Wählen Sie **Remove**.

AWS CLI

Ersetzen Sie `my-cluster` durch den Namen Ihres Clusters und führen Sie dann den folgenden Befehl aus.

```
aws eks delete-addon --cluster-name my-cluster --addon-name
netapp_trident-operator --preserve
```

eksctl

Der folgende Befehl deinstalliert das Trident EKS add-on:

```
eksctl delete addon --cluster K8s-arm --name netapp_trident-operator
```

Konfigurieren einer Speicherklasse

Der "Kubernetes StorageClass-Objekt" identifiziert einen Provisioner und weist den Provisioner an, wie Volumes bereitgestellt werden sollen. In diesem Abschnitt wird gezeigt, wie Sie ein Kubernetes-StorageClass-Objekt konfigurieren, das Trident als Provisioner angibt.

Erstellen Sie ein StorageClass-Objekt

Wenn Sie eine StorageClass für FSx for ONTAP erstellen, erstellt Trident automatisch die Backend-Konfiguration.

HINWEIS	Wenn Sie das Speicher-Backend manuell konfigurieren möchten, lesen Sie bitte den Erstellen Sie eine Kubernetes StorageClass ohne automatische Backend-Konfiguration Abschnitt, um zu erfahren, wie Sie das Trident-Backend und die Speicherklasse separat erstellen.

Geben Sie die erforderlichen StorageClass-Parameter an

Die folgenden drei Parameter müssen beim Erstellen einer StorageClass definiert werden:

Parameter	Erforderlich	Typ	Beschreibung
`fsxFilesystemID`	Ja	Zeichenkette	FSx für NetApp ONTAP filesystem ID
`storageDriverName`	Ja	Zeichenkette	Trident-Speichertreiber (zum Beispiel, `ontap-nas` oder `ontap-san`)
`credentialsName`	Ja	Zeichenkette	Name des Kubernetes-Secrets, das die FSx for ONTAP Anmeldeinformationen enthält

Optionale Parameter angeben

Sie können optionale Backend-Parameter über die StorageClass übergeben. Definieren Sie alle optionalen Werte als Zeichenketten im StorageClass `parameters` Abschnitt. Eine vollständige Liste der Backend-Parameter finden Sie unter: "FSx für NetApp ONTAP Backend-Konfiguration".

Beispiel-StorageClass-Konfigurationsdateien.

Das folgende Beispiel zeigt eine StorageClass, die eine automatische Backend-Konfiguration auslöst.

YAML

```yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-fsx-demo
  annotations:
    description: "Demo StorageClass for FSx for NetApp ONTAP"
provisioner: csi.trident.netapp.io
parameters:
  fsxFilesystemID: "fs-0abc123"
  storageDriverName: "ontap-nas"
  credentialsName: trident-fsx-credentials
allowVolumeExpansion: true
reclaimPolicy: Delete
volumeBindingMode: Immediate
```

JSON

```json
{
  "apiVersion": "storage.k8s.io/v1",
  "kind": "StorageClass",
  "metadata": {
    "name": "ontap-fsx-demo",
    "annotations": {
      "description": "Demo StorageClass for FSx for NetApp ONTAP"
    }
  },
  "provisioner": "csi.trident.netapp.io",
  "parameters": {
    "fsxFilesystemID": "fs-0abc123",
    "storageDriverName": "ontap-nas",
    "credentialsName": "trident-fsx-credentials"
  },
  "allowVolumeExpansion": true,
  "reclaimPolicy": "Delete",
  "volumeBindingMode": "Immediate"
}
```

Erstelle die StorageClass

Sobald Sie Ihre Konfigurationsdatei erstellt haben, führen Sie den folgenden Befehl aus, um die Speicherklasse zu erstellen.

```
kubectl create -f storage-class-ontapnas.yaml
```

Sie sollten nun eine **basic-csi** Storage-Class sowohl in Kubernetes als auch in Trident sehen, und Trident sollte die Pools im Backend erkannt haben.

```
kubectl get sc basic-csi
```

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

Nachdem Sie die StorageClass angewendet haben, erstellt Trident das Backend automatisch. Sie können dann PersistentVolumeClaims erstellen, die auf diese StorageClass verweisen.

Überprüfen Sie den Status der Backend-Konfiguration

Trident speichert das Ergebnis der Backend-Erstellung in StorageClass-Annotationen.

Anmerkung	Beschreibung
`trident.netapp.io/configuratorStatus`	Konfigurationsergebnis (`Success` or `Failure`)
`trident.netapp.io/configuratorMessage`	Detaillierte Status- oder Fehlermeldung
`trident.netapp.io/configuratorName`	Name der internen Konfigurator-Ressource
`trident.netapp.io/managed`	Gibt an, dass die StorageClass von Trident verwaltet wird
`trident.netapp.io/additionalStoragePools`	Für dieses Backend erstellte Speicherpools

Um den Status zu überprüfen, führen Sie Folgendes aus:

```
kubectl get storageclass ontap-fsx-demo -o yaml
```

Bestätigen Sie, dass `trident.netapp.io/configuratorStatus` auf `Success` gesetzt ist. Wenn der Wert `Failure` ist, überprüfen Sie `trident.netapp.io/configuratorMessage` auf den Fehler.

Fügen Sie zusätzliche FSxN-Dateisysteme hinzu

Wenn Sie zusätzliche Speicherkapazität benötigen, während Sie weiterhin dieselbe StorageClass verwenden, fügen Sie zusätzliche FSxN-Dateisystem-IDs hinzu.

Bearbeiten Sie die StorageClass und fügen Sie die folgende Anmerkung hinzu:

```
metadata:
  annotations:
    trident.netapp.io/additionalFsxNFileSystemID: '["fs-
xxxxxxxxxxxxxxxxx"]'
```

Nachdem Sie die Änderung angewendet haben, aktualisiert Trident die Backend-Konfiguration und die StorageClass-Annotationen.

Betriebliche Überlegungen und Einschränkungen

- Das Löschen einer StorageClass, die als automatische Backend-Konfiguration verwendet wird, löscht in der Regel auch das zugehörige Trident-Backend. Dies kann die Speicherkonnektivität beeinträchtigen und laufende Workloads unterbrechen. Prüfen Sie die Auswirkungen, bevor Sie eine verwaltete StorageClass löschen.
- Die automatische Backend-Konfiguration wird nur für AWS FSx für NetApp ONTAP unterstützt.

Erstellen Sie eine Kubernetes StorageClass ohne automatische Backend-Konfiguration

Wenn Sie das Trident Backend und StorageClass separat erstellen möchten, folgen Sie diesen Schritten.

Verstehen Sie, wie die automatische Backend-Konfiguration funktioniert

Trident leitet die Backend-Konfiguration aus der StorageClass-Definition ab. Wenn Sie die StorageClass anwenden, validiert Trident die erforderlichen Parameter, erstellt das Backend und versieht die StorageClass mit dem Status.

Trident erstellt die VolumeSnapshotClass nur einmal. Trident verwendet dieselbe VolumeSnapshotClass für nachfolgende StorageClasses wieder.

Erstelle das Trident Backend

Um ein Trident-Backend zu erstellen, benötigen Sie eine Konfigurationsdatei im JSON- oder YAML-Format. Die Datei muss den gewünschten Speichertyp (NAS oder SAN), das Dateisystem, die SVM, von der Sie es beziehen möchten, und die Authentifizierungsmethode angeben. Das folgende Beispiel zeigt, wie Sie NAS-basierten Speicher definieren und mithilfe eines AWS-Secrets die Anmeldeinformationen für die SVM speichern, die Sie verwenden möchten:

YAML

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-nas
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-nas
  backendName: tbc-ontap-nas
  svm: svm-name
  aws:
    fsxFilesystemID: fs-xxxxxxxxxx
  credentials:
    name: "arn:aws:secretsmanager:us-west-2:xxxxxxxx:secret:secret-
name"
    type: awsarn
```

JSON

```
{
  "apiVersion": "trident.netapp.io/v1",
  "kind": "TridentBackendConfig",
  "metadata": {
    "name": "backend-tbc-ontap-nas"
    "namespace": "trident"
  },
  "spec": {
    "version": 1,
    "storageDriverName": "ontap-nas",
    "backendName": "tbc-ontap-nas",
    "svm": "svm-name",
    "aws": {
      "fsxFilesystemID": "fs-xxxxxxxxxx"
    },
    "managementLIF": null,
    "credentials": {
      "name": "arn:aws:secretsmanager:us-west-2:xxxxxxxx:secret:secret-
name",
      "type": "awsarn"
    }
  }
}
```

FSx for ONTAP-Treiberdetails

Sie können Trident mit Amazon FSx für NetApp ONTAP mithilfe der folgenden Treiber integrieren:

Treibername	Beschreibung
`ontap-san`	Jedes bereitgestellte PV ist eine LUN innerhalb eines eigenen Amazon FSx for NetApp ONTAP-Volumes. Empfohlen für Blockspeicher.
`ontap-nas`	Jedes bereitgestellte PV ist ein vollständiges Amazon FSx for NetApp ONTAP Volume. Empfohlen für NFS und SMB.
`ontap-san-economy`	Jedes bereitgestellte PV ist eine LUN mit einer konfigurierbaren Anzahl von LUNs pro Amazon FSx for NetApp ONTAP volume.
`ontap-nas-economy`	Jedes bereitgestellte PV ist ein Qtree, wobei die Anzahl der Qtrees pro Amazon FSx for NetApp ONTAP Volume konfigurierbar ist.
`ontap-nas-flexgroup`	Jedes bereitgestellte PV ist ein vollständiges Amazon FSx for NetApp ONTAP FlexGroup Volume.

Weitere Informationen zum Treiber finden Sie unter "NAS-Treiber" und "SAN-Treiber".

Das Backend erstellen

Nach dem Erstellen der Konfigurationsdatei führen Sie die folgenden Befehle aus, um die Trident Backend-Konfiguration (TBC) zu erstellen und zu validieren:

- Erstellen Sie eine Trident-Backend-Konfiguration (TBC) aus einer yaml-Datei und führen Sie den folgenden Befehl aus:

```
kubectl create -f backendconfig.yaml -n trident
```

```
tridentbackendconfig.trident.netapp.io/backend-tbc-ontap-nas created
```

- Überprüfen Sie, ob die Trident-Backend-Konfiguration (TBC) erfolgreich erstellt wurde:

```
Kubectl get tbc -n trident
```

```
NAME                           BACKEND NAME          BACKEND UUID
PHASE   STATUS

backend-tbc-ontap-nas          tbc-ontap-nas         933e0071-66ce-4324-
b9ff-f96d916ac5e9   Bound   Success
```

Weitere Informationen zu anderen Konfigurationsoptionen finden Sie im [Backend-advanced-configuration-and-examples] Abschnitt unten.

Konfigurieren einer Storage Class ohne automatische Backend-Konfiguration

Nachfolgend finden Sie Beispiele für Speicherklassenkonfigurationen zur Verwendung mit Trident und FSx für ONTAP.

Speicherklasse für NFS

Sie können dieses Beispiel verwenden, um StorageClass für Volumes mit NFS einzurichten (eine vollständige Liste der Attribute finden Sie im Abschnitt „Trident-Attribute“ weiter unten):

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-gold
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-nas"
  provisioningType: "thin"
  snapshots: "true"
```

Speicherklasse für iSCSI

Verwenden Sie dieses Beispiel, um StorageClass für Volumes mit iSCSI einzurichten:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-gold
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
  provisioningType: "thin"
  snapshots: "true"
```

Speicherklasse mit NFSv3 und AWS Bottlerocket

Um NFSv3-Volumes auf AWS Bottlerocket bereitzustellen, fügen Sie die erforderlichen `mountOptions` zur Speicherklasse hinzu:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-gold
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-nas"
  media: "ssd"
  provisioningType: "thin"
  snapshots: "true"
mountOptions:
  - nfsvers=3
  - nolock
```

Trident StorageClass-Attribute

Diese Parameter legen fest, welche von Trident verwalteten Speicherpools zur Bereitstellung von Volumes eines bestimmten Typs verwendet werden sollen.

Attribut	Typ	Werte	Angebot	Anfrage	Unterstützt von
Medien[1]	Zeichenkette	hdd, hybrid, ssd	Pool enthält Medien dieses Typs; hybrid bedeutet beides	Medientyp angegeben	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, solidfire-san
provisioningType	Zeichenkette	dünn, dick	Pool unterstützt diese Bereitstellungsmethode	Bereitstellungsmethode angegeben	dick: alle ontap; dünn: alle ontap & solidfire-san
backendType	Zeichenkette	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, solidfire-san, azure-netapp-files, ontap-san-economy	Pool gehört zu dieser Art von Backend	Backend angegeben	Alle Treiber
Momentaufnahmen	bool	wahr, falsch	Pool unterstützt Volumes mit Snapshots	Volume mit aktivierten Snapshots	ontap-nas, ontap-san, solidfire-san
Klone	bool	wahr, falsch	Pool unterstützt das Klonen von Volumes	Volume mit aktivierten Klonen	ontap-nas, ontap-san, solidfire-san

Attribut	Typ	Werte	Angebot	Anfrage	Unterstützt von
Verschlüsselung	bool	wahr, falsch	Pool unterstützt verschlüsselte Volumes	Volume mit aktivierter Verschlüsselung	ontap-nas, ontap-nas-economy, ontap-nas-flexgroups, ontap-san
IOPS	int	positive ganze Zahl	Pool ist in der Lage, IOPS in diesem Bereich zu garantieren	Volume garantiert diese IOPS	solidfire-san

[1]: Wird von ONTAP Select oder FSx für ONTAP-Systeme nicht unterstützt

Siehe "Kubernetes- und Trident-Objekte" für Details dazu, wie Storage-Klassen mit dem `PersistentVolumeClaim` interagieren und Parameter zur Steuerung, wie Trident Volumes bereitstellt.

Erstellen Sie die Speicherklasse

Sobald Sie die StorageClass konfiguriert haben, können Sie sie in Kubernetes erstellen.

Schritte

1. Dies ist ein Kubernetes-Objekt, daher verwenden Sie `kubectl`, um es in Kubernetes zu erstellen.

```
kubectl create -f storage-class-ontapnas.yaml
```

2. Sie sollten nun eine **basic-csi** Storage-Class sowohl in Kubernetes als auch in Trident sehen, und Trident sollte die Pools im Backend erkannt haben.

```
kubectl get sc basic-csi
```

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

SMB-Volumen bereitstellen

Sie können SMB-Volumes mithilfe des `ontap-nas` Treibers bereitstellen. Dazu müssen Sie jedoch die folgenden Schritte ausführen: "Bereiten Sie die Bereitstellung von SMB-Volumes vor".

Erweiterte Backend-Konfiguration und Beispiele

Siehe die folgende Tabelle für die Backend-Konfigurationsoptionen:

Parameter	Beschreibung	Beispiel
`version`		Immer 1

Parameter	Beschreibung	Beispiel
`storageDriverName`	Name des Speichertreibers	`ontap-nas`, `ontap-nas-economy`, `ontap-nas-flexgroup`, `ontap-san`, `ontap-san-economy`
`backendName`	Benutzerdefinierter Name oder das Speicher-Backend	Treibername + "_" + dataLIF
`managementLIF`	IP-Adresse eines Clusters oder einer SVM-Management-LIF. Ein vollqualifizierter Domänenname (FQDN) kann angegeben werden. Kann so eingestellt werden, dass IPv6-Adressen verwendet werden, wenn Trident mit dem IPv6-Flag installiert wurde. IPv6-Adressen müssen in eckigen Klammern angegeben werden, zum Beispiel [28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]. Wenn Sie die `fsxFilesystemID` unter dem `aws` Feld angeben, müssen Sie die `managementLIF` nicht angeben, da Trident die SVM `managementLIF` Informationen von AWS abruft. Sie müssen daher Anmeldeinformationen für einen Benutzer unter der SVM (zum Beispiel: vsadmin) angeben, und der Benutzer muss die `vsadmin` Rolle haben.	"10.0.0.1", "[2001:1234:abcd::fefe]"

Parameter	Beschreibung	Beispiel
`dataLIF`	IP-Adresse des Protokoll-LIF. **ONTAP NAS-Treiber**: NetApp empfiehlt, dataLIF anzugeben. Falls nicht angegeben, ruft Trident dataLIFs vom SVM ab. Ein vollqualifizierter Domainname (FQDN) kann für die NFS-Mount-Operationen angegeben werden, wodurch die Erstellung eines Round-Robin-DNS zur Lastverteilung über mehrere dataLIFs ermöglicht wird. Kann nach der Ersteinrichtung geändert werden. **ONTAP SAN-Treiber**: Für iSCSI nicht angeben. Trident verwendet ONTAP Selective LUN Map, um die für den Aufbau einer Multipath-Sitzung benötigten iSCSI-LIFs zu ermitteln. Eine Warnung wird ausgegeben, wenn dataLIF explizit definiert ist. Kann so eingestellt werden, dass IPv6-Adressen verwendet werden, wenn Trident mit dem IPv6-Flag installiert wurde. IPv6-Adressen müssen in eckigen Klammern angegeben werden, zum Beispiel [28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555].	
`autoExportPolicy`	Automatische Erstellung und Aktualisierung von Exportrichtlinien aktivieren [Boolean]. Mit den Optionen `autoExportPolicy` und `autoExportCIDRs` kann Trident Exportrichtlinien automatisch verwalten.	`false`
`autoExportCIDRs`	Liste der CIDRs, anhand derer die Node-IPs von Kubernetes gefiltert werden, wenn `autoExportPolicy` aktiviert ist. Mit den Optionen `autoExportPolicy` und `autoExportCIDRs` kann Trident Exportrichtlinien automatisch verwalten.	"["0.0.0.0/0", "::/0"]"
`labels`	Satz beliebiger JSON-formatierter Bezeichnungen, die auf Volumes angewendet werden sollen	""

Parameter	Beschreibung	Beispiel
`clientCertificate`	Base64-kodierter Wert des Clientzertifikats. Wird für zertifikatbasierte Authentifizierung verwendet	""
`clientPrivateKey`	Base64-kodierter Wert des privaten Client-Schlüssels. Wird für zertifikatbasierte Authentifizierung verwendet	""
`trustedCACertificate`	Base64-kodierter Wert des vertrauenswürdigen CA-Zertifikats. Optional. Wird für zertifikatbasierte Authentifizierung verwendet.	""
`username`	Benutzername für die Verbindung zum Cluster oder zur SVM. Wird für die anmeldeinformationsbasierte Authentifizierung verwendet. Zum Beispiel vsadmin.	
`password`	Passwort zur Verbindung mit dem Cluster oder der SVM. Wird für die anmeldeinformationsbasierte Authentifizierung verwendet.	
`svm`	Zu verwendende Storage Virtual Machine	Wird abgeleitet, wenn ein SVM managementLIF angegeben ist.
`storagePrefix`	Präfix, das beim Bereitstellen neuer Volumes in der SVM verwendet wird. Kann nach der Erstellung nicht mehr geändert werden. Um diesen Parameter zu aktualisieren, müssen Sie ein neues Backend erstellen.	`trident`
`limitAggregateUsage`	**Nicht für Amazon FSx für NetApp ONTAP angeben.** Die bereitgestellten `fsxadmin` und `vsadmin` enthalten nicht die erforderlichen Berechtigungen, um die aggregierte Nutzung abzurufen und sie mit Trident zu begrenzen.	Nicht verwenden.

Parameter	Beschreibung	Beispiel
`limitVolumeSize`	Die Bereitstellung schlägt fehl, wenn die angeforderte Volume-Größe diesen Wert überschreitet. Außerdem wird die maximale Größe der von ihr verwalteten Volumes für qtrees und LUNs beschränkt, und die `qtreesPerFlexvol` Option ermöglicht die Anpassung der maximalen Anzahl von qtrees pro FlexVol volume.	"" (wird nicht standardmäßig erzwungen)
`lunsPerFlexvol`	Maximale LUNs pro FlexVol volume, muss im Bereich [50, 200] liegen. Nur SAN.	“100”
`debugTraceFlags`	Debug-Flags zur Verwendung bei der Fehlersuche. Beispiel, {"api":false, "method":true} Verwenden Sie `debugTraceFlags` nicht, es sei denn, Sie führen eine Fehlersuche durch und benötigen eine detaillierte Protokollausgabe.	null
`nfsMountOptions`	Kommagetrennte Liste von NFS-Mountoptionen. Die Mountoptionen für persistente Kubernetes-Volumes werden normalerweise in Speicherklassen angegeben, aber wenn keine Mountoptionen in einer Speicherklasse angegeben sind, verwendet Trident die Mountoptionen, die in der Konfigurationsdatei des Storage-Backends angegeben sind. Wenn weder in der Speicherklasse noch in der Konfigurationsdatei Mountoptionen angegeben sind, setzt Trident keine Mountoptionen für das zugehörige persistente Volume.	""
`nasType`	Konfigurieren Sie die Erstellung von NFS- oder SMB-Volumes. Optionen sind `nfs`, `smb` oder null. **Muss auf `smb` gesetzt werden für SMB-Volumes.** Die Einstellung auf null verwendet standardmäßig NFS-Volumes.	`nfs`
`qtreesPerFlexvol`	Maximale Qtrees pro FlexVol volume, muss im Bereich [50, 300] liegen	`"200"`

Parameter	Beschreibung	Beispiel
`smbShare`	Sie können entweder den Namen einer SMB-Freigabe angeben, die mit der Microsoft Management Console oder ONTAP CLI erstellt wurde, oder einen Namen, damit Trident die SMB-Freigabe erstellen kann. Dieser Parameter ist für Amazon FSx for ONTAP Backends erforderlich.	`smb-share`
`useREST`	Boolescher Parameter zur Verwendung von ONTAP REST APIs. Wenn auf `true` gesetzt, verwendet Trident ONTAP REST APIs, um mit dem Backend zu kommunizieren. Diese Funktion erfordert ONTAP 9.11.1 und höher. Zusätzlich muss die verwendete ONTAP-Anmelderolle Zugriff auf die `ontap` Anwendung haben. Dies wird durch die vordefinierten `vsadmin` und `cluster-admin` Rollen erfüllt.	`false`
`aws`	Sie können Folgendes in der Konfigurationsdatei für AWS FSx for ONTAP angeben: - `fsxFilesystemID`: Geben Sie die ID des AWS FSx-Dateisystems an. - `apiRegion`: Name der AWS API-Region. - `apikey`: AWS API-Schlüssel. - `secretKey`: Geheimer AWS-Schlüssel.	"" "" ""
`credentials`	Geben Sie die FSx SVM-Anmeldeinformationen an, die im AWS Secrets Manager gespeichert werden sollen. - `name`: Amazon Resource Name (ARN) des Secrets, das die Anmeldeinformationen der SVM enthält. - `type`: Auf `awsarn` setzen. Weitere Informationen sind unter "Erstellen Sie ein AWS Secrets Manager secret" verfügbar.	

Backend-Konfigurationsoptionen für die Bereitstellung von Volumes

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

Parameter	Beschreibung	Standard
`spaceAllocation`	Speicherplatzzuweisung für LUNs	`true`
`spaceReserve`	Platzreservierungsmodus; "none" (dünn) oder "volume" (dick)	`none`
`snapshotPolicy`	Zu verwendende Snapshot-Richtlinie	`none`
`qosPolicy`	QoS-Richtliniengruppe, die für erstellte Volumes zugewiesen werden soll. Wählen Sie eine der qosPolicy oder adaptiveQosPolicy pro Speicherpool oder Backend. Die Verwendung von QoS-Richtliniengruppen mit Trident erfordert ONTAP 9.8 oder höher. Sie sollten eine nicht gemeinsam genutzte QoS-Richtliniengruppe verwenden und sicherstellen, dass die Richtliniengruppe auf jedes einzelne Element angewendet wird. Eine gemeinsam genutzte QoS-Richtliniengruppe erzwingt die Obergrenze für den Gesamtdurchsatz aller Workloads.	""
`adaptiveQosPolicy`	Adaptive QoS-Richtliniengruppe zur Zuweisung für erstellte Volumes. Wählen Sie eine der qosPolicy oder adaptiveQosPolicy pro Speicherpool oder Backend. Wird von ontap-nas-economy nicht unterstützt.	""
`snapshotReserve`	Prozentsatz des für Snapshots reservierten Volumens "0"	Wenn `snapshotPolicy` ist `none`, `else` ""
`splitOnClone`	Trennen Sie einen Klon bei seiner Erstellung von seinem übergeordneten Objekt	`false`
`encryption`	Aktivieren Sie NetApp Volume Encryption (NVE) auf dem neuen Volume; Standard ist `false`. NVE muss auf dem Cluster lizenziert und aktiviert sein, um diese Option zu verwenden. Wenn NAE im Backend aktiviert ist, wird jedes in Trident bereitgestellte Volume NAE-aktiviert. Weitere Informationen finden Sie unter: "Wie Trident mit NVE und NAE zusammenarbeitet".	`false`

Parameter	Beschreibung	Standard
`luksEncryption`	LUKS-Verschlüsselung aktivieren. Siehe "Verwenden Sie Linux Unified Key Setup (LUKS)". Nur SAN.	""
`tieringPolicy`	zu verwendende Tiering-Richtlinie `none`	
`unixPermissions`	Modus für neue Volumes. **Für SMB-Volumes leer lassen.**	""
`securityStyle`	Sicherheitsstil für neue Volumes. NFS unterstützt `mixed` und `unix` Sicherheitsstile. SMB unterstützt `mixed` und `ntfs` Sicherheitsstile.	NFS-Standard ist `unix`. SMB-Standard ist `ntfs`.

PVC konfigurieren

Dieser Abschnitt enthält Anweisungen zum Erstellen eines PersistentVolumeClaim (PVC), der die konfigurierte Kubernetes-StorageClass verwendet, um ein PV anzufordern. Im Erfolgsfall kann das PV anschließend in einen Pod eingebunden werden.

PVC erstellen

Ein "*PersistentVolumeClaim*" (PVC) ist eine Anfrage für den Zugriff auf das PersistentVolume im Cluster. Die PVC kann so konfiguriert werden, dass sie Speicherplatz einer bestimmten Größe oder eines bestimmten Zugriffsmodus anfordert. Mithilfe der zugehörigen StorageClass kann der Clusteradministrator mehr als nur PersistentVolume-Größe und Zugriffsmodus steuern – zum Beispiel Leistung oder Servicelevel.

Nachdem Sie das Trident-Backend und die StorageClass erstellt haben, können Sie ein PVC erstellen. Sobald das PVC erstellt ist, können Sie das Volume in einem Pod einbinden.

Beispielmanifeste

Die folgenden Beispiele zeigen grundlegende PVC-Konfigurationsoptionen.

PVC mit RWX-Zugriff

Dieses Beispiel zeigt eine einfache PVC mit RWX-Zugriff, die mit einer StorageClass namens `basic-csi` verknüpft ist.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc-storage
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: ontap-gold
```

PVC mit iSCSI-Beispiel

Dieses Beispiel zeigt eine einfache PVC für iSCSI mit RWO-Zugriff, die einer StorageClass namens `protection-gold` zugeordnet ist.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc-san
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: protection-gold
```

PVC erstellen

Schritte

1. Erstellen Sie das PVC.

   ```
   kubectl create -f pvc.yaml
   ```

2. Überprüfen Sie den PVC-Status.

   ```
   kubectl get pvc
   ```

```
NAME         STATUS VOLUME     CAPACITY ACCESS MODES STORAGECLASS AGE
pvc-storage Bound  pv-name 2Gi      RWO                           5m
```

Siehe "Kubernetes- und Trident-Objekte" für Details dazu, wie Storage-Klassen mit dem `PersistentVolumeClaim` interagieren und Parameter zur Steuerung, wie Trident Volumes bereitstellt.

Eine Anwendung bereitstellen

Sobald die Storage Class und die PVC erstellt sind, können Sie das PV an einen Pod anhängen. In diesem Abschnitt finden Sie den Beispielbefehl und die Konfiguration, um das PV an einen Pod anzuhängen.

Eine Beispielanwendung bereitstellen

Schritte

1. Mountieren Sie das Volume in einem Pod.

   ```
   kubectl create -f pv-pod.yaml
   ```

   Diese Beispiele zeigen grundlegende Konfigurationen, um das PVC an ein Pod anzuhängen:
   Grundkonfiguration:

   ```
   kind: Pod
   apiVersion: v1
   metadata:
     name: pv-pod
   spec:
     volumes:
       - name: pv-storage
         persistentVolumeClaim:
          claimName: basic
     containers:
       - name: pv-container
         image: nginx
         ports:
           - containerPort: 80
             name: "http-server"
         volumeMounts:
           - mountPath: "/my/mount/path"
             name: pv-storage
   ```

   HINWEIS Sie können den Fortschritt mit `kubectl get pod --watch` überwachen.

2. Überprüfen Sie, ob das Volume auf `/my/mount/path` eingebunden ist.

```
kubectl exec -it pv-pod -- df -h /my/mount/path
```

```
Filesystem                                                      Size
Used  Avail Use%  Mounted on
192.168.188.78:/trident_pvc_ae45ed05_3ace_4e7c_9080_d2a83ae03d06  1.1G
320K  1.0G   1%   /my/mount/path
```

Sie können den Pod nun löschen. Die Pod-Anwendung existiert dann nicht mehr, aber das Volume bleibt erhalten.

```
kubectl delete pod pv-pod
```

Konfigurieren Sie das Trident EKS add-on auf einem EKS Cluster

NetApp Trident vereinfacht die Verwaltung von Amazon FSx for NetApp ONTAP Storage in Kubernetes, sodass sich Ihre Entwickler und Administratoren auf die Anwendungsbereitstellung konzentrieren können. Das NetApp Trident EKS Add-on enthält die neuesten Sicherheitspatches, Fehlerbehebungen und ist von AWS für die Zusammenarbeit mit Amazon EKS validiert. Das EKS-Add-on ermöglicht es Ihnen, konsequent sicherzustellen, dass Ihre Amazon EKS-Cluster sicher und stabil sind, und reduziert den Arbeitsaufwand, den Sie für die Installation, Konfiguration und Aktualisierung von Add-ons aufwenden müssen.

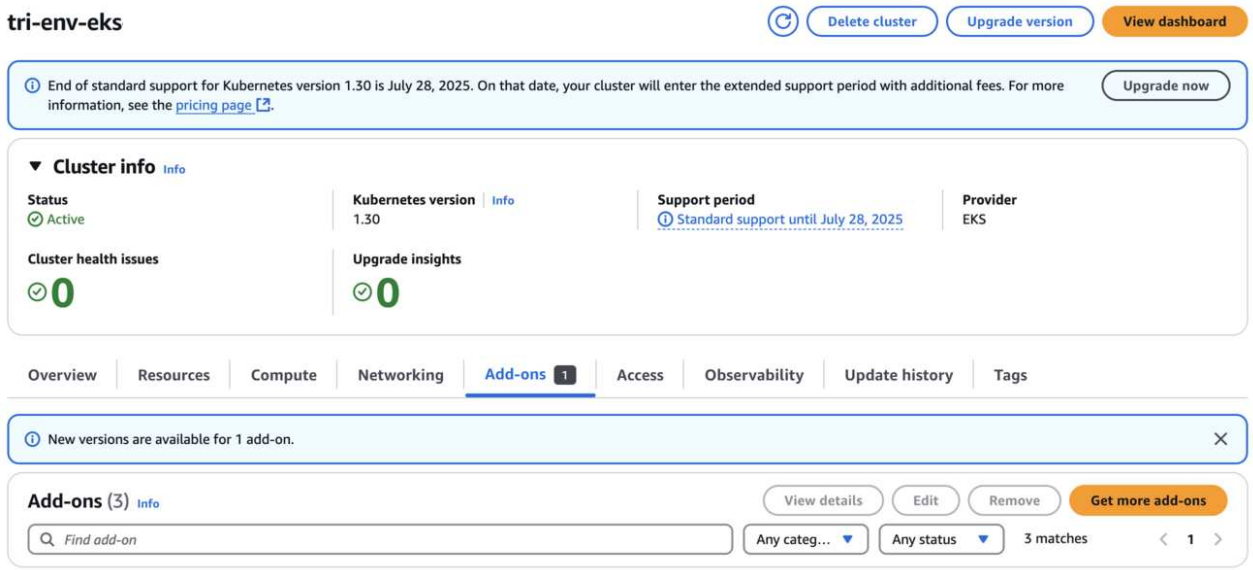
Voraussetzungen

Stellen Sie sicher, dass Sie Folgendes haben, bevor Sie das Trident-Add-on für AWS EKS konfigurieren:

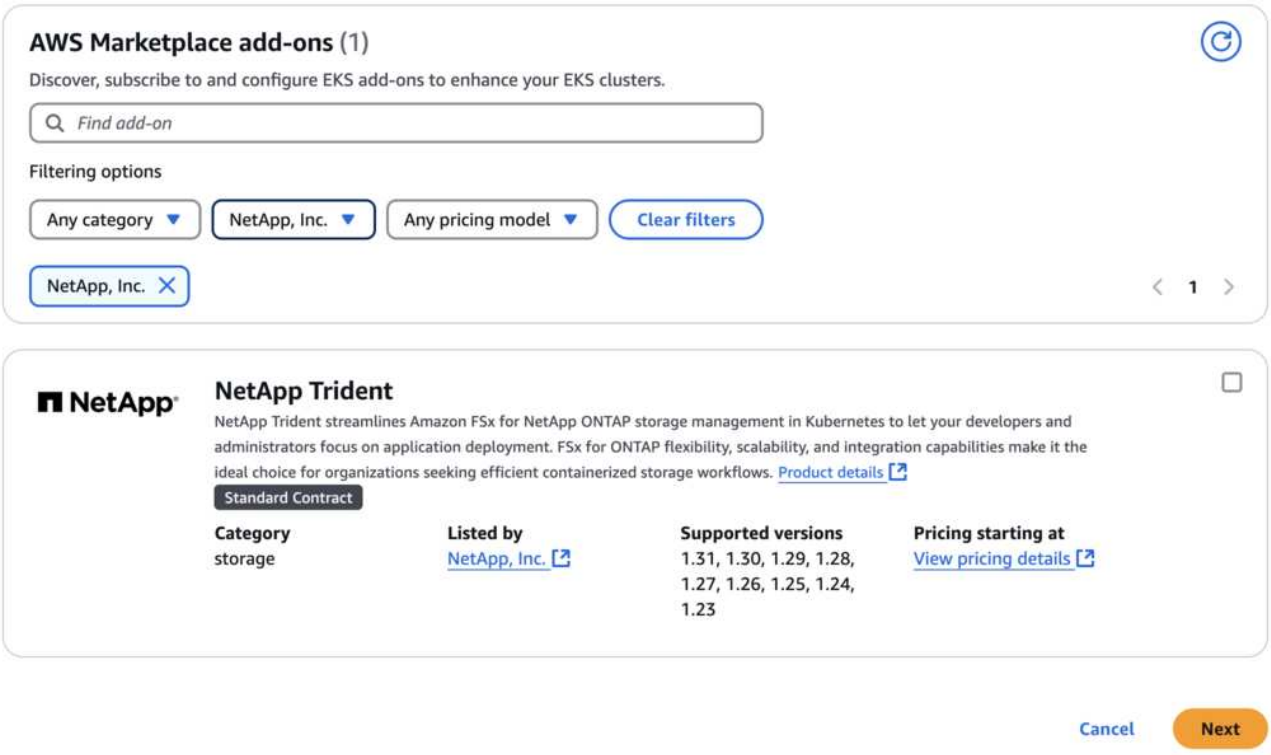
- Ein Amazon EKS-Clusterkonto mit Berechtigungen zur Verwendung von Add-ons. Siehe "Amazon EKS Add-ons".
- AWS-Berechtigungen für den AWS Marketplace:
  `"aws-marketplace:ViewSubscriptions",`
  `"aws-marketplace:Subscribe",`
  `"aws-marketplace:Unsubscribe`
- AMI type: Amazon Linux 2 (AL2_x86_64) oder Amazon Linux 2 Arm (AL2_ARM_64)
- Knotentyp: AMD oder ARM
- Ein bestehendes Amazon FSx for NetApp ONTAP file system

Schritte

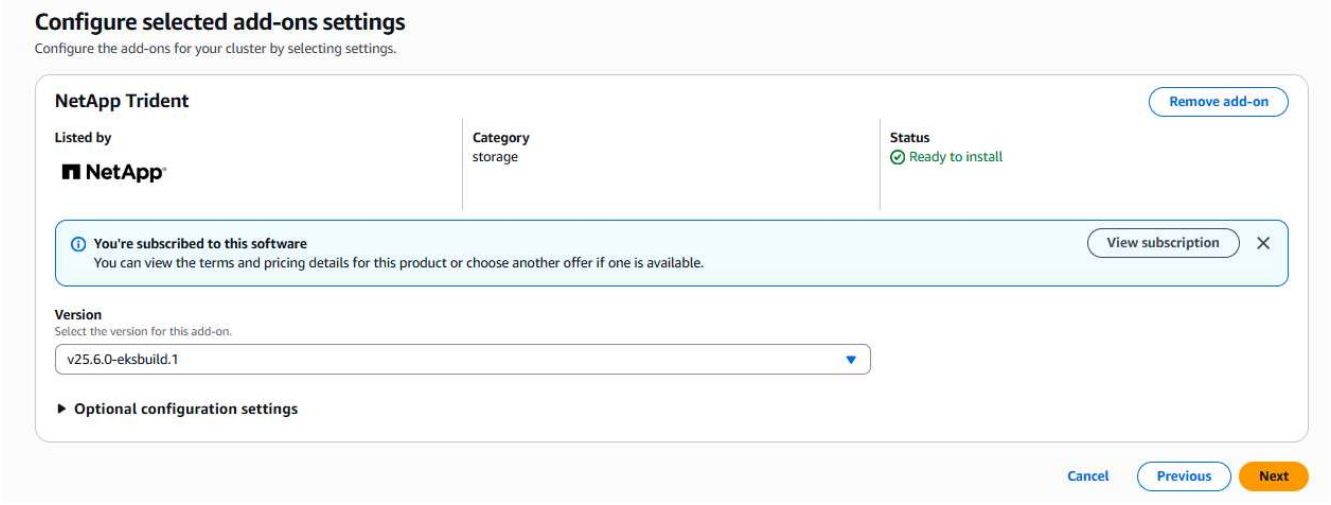
1. Stellen Sie sicher, dass Sie eine IAM-Rolle und ein AWS-Secret erstellen, damit EKS-Pods auf AWS-Ressourcen zugreifen können. Anweisungen dazu finden Sie unter "Erstellen Sie eine IAM-Rolle und ein AWS-Geheimnis".
2. Navigieren Sie in Ihrem EKS Kubernetes-Cluster zur Registerkarte **Add-ons**.



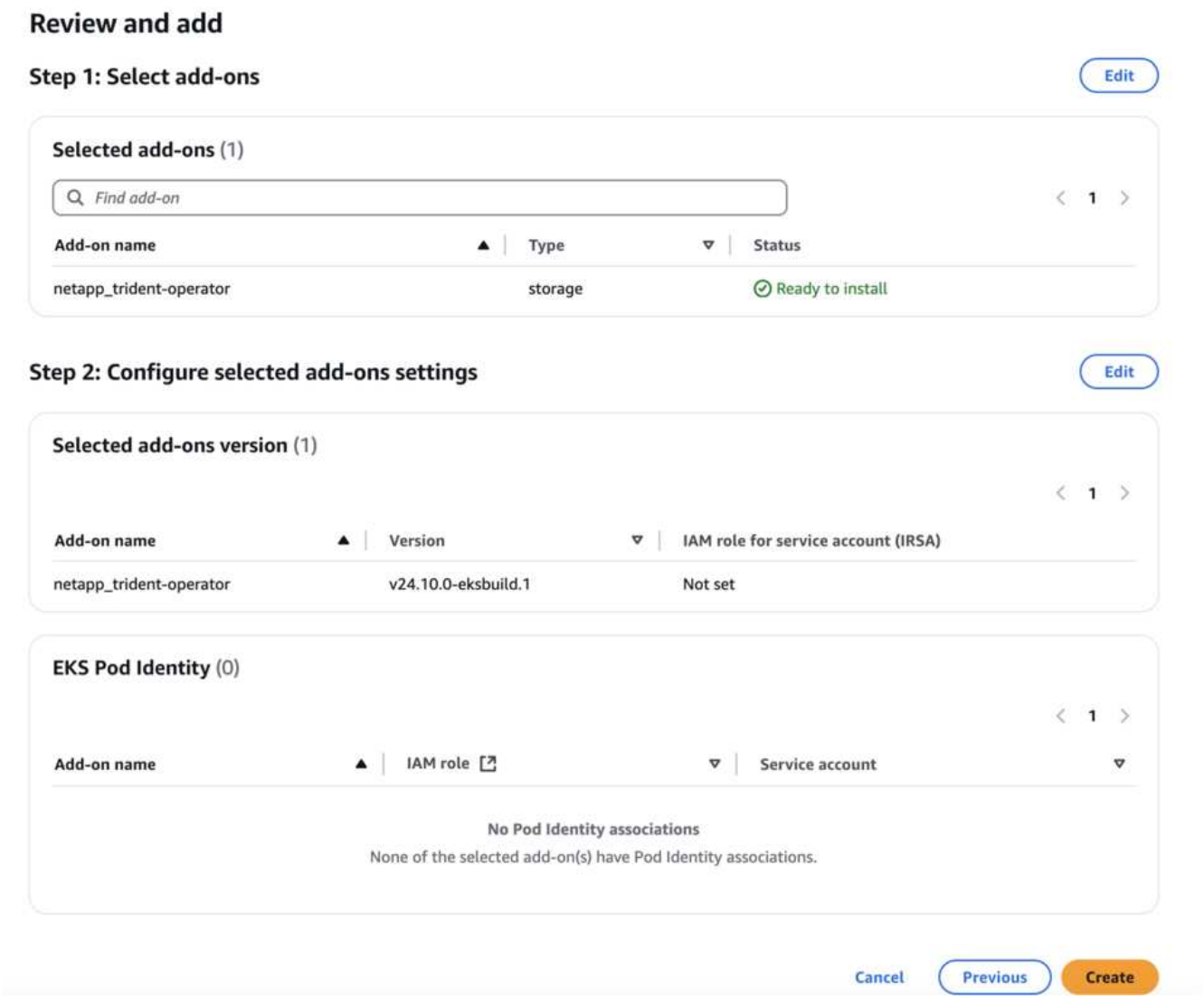
3. Gehen Sie zu **AWS Marketplace add-ons** und wählen Sie die Kategorie *storage*.



4. Suchen Sie **NetApp Trident** und aktivieren Sie das Kontrollkästchen für das Trident-Add-on, und klicken Sie auf **Weiter**.
5. Wählen Sie die gewünschte Version des Add-on.

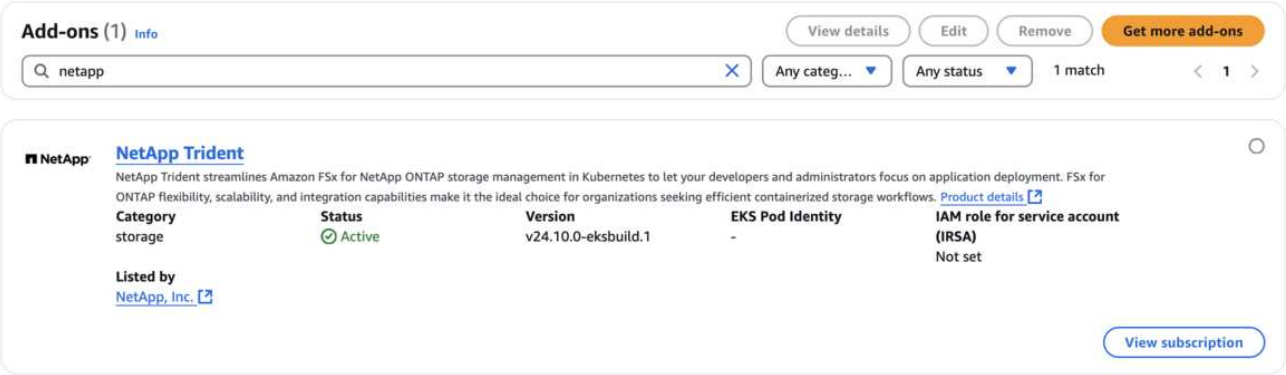


6. Konfigurieren Sie die erforderlichen Add-on-Einstellungen.



7. Wenn Sie IRSA (IAM roles for service account) verwenden, beachten Sie die zusätzlichen Konfigurationsschritte "hier,".
8. Wählen Sie **Create**.

9. Überprüfen Sie, ob der Status des Add-ons *Active* ist.



10. Führen Sie den folgenden Befehl aus, um zu überprüfen, ob Trident ordnungsgemäß auf dem Cluster installiert ist:

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

11. Fahren Sie mit der Einrichtung fort und konfigurieren Sie das Storage-Backend. Weitere Informationen finden Sie unter "Konfigurieren Sie das Storage-Backend".

Installieren/Deinstallieren des Trident EKS Add-ons über die CLI

Installieren Sie das NetApp Trident EKS-Add-on mithilfe der CLI:

Der folgende Beispielbefehl installiert das Trident EKS-Add-on:
`eksctl create addon --cluster clusterName --name netapp_trident-operator --version v25.6.0-eksbuild.1` (mit einer dedizierten Version)

Der folgende Beispielbefehl installiert das Trident EKS-Add-on Version 25.6.1:
`eksctl create addon --cluster clusterName --name netapp_trident-operator --version v25.6.1-eksbuild.1` (mit einer dedizierten Version)

Der folgende Beispielbefehl installiert das Trident EKS-Add-on Version 25.6.2:
`eksctl create addon --cluster clusterName --name netapp_trident-operator --version v25.6.2-eksbuild.1` (mit einer dedizierten Version)

Deinstallieren Sie das NetApp Trident EKS add-on über die CLI:

Der folgende Befehl deinstalliert das Trident EKS add-on:

```
eksctl delete addon --cluster K8s-arm --name netapp_trident-operator
```

Backends mit kubectl erstellen

Ein Backend definiert die Beziehung zwischen Trident und einem Speichersystem. Es gibt Trident vor, wie Trident mit diesem Speichersystem kommuniziert und wie Trident Volumes daraus bereitstellen soll. Nachdem Trident installiert wurde, besteht der nächste Schritt darin, ein Backend zu erstellen. Die `TridentBackendConfig` Custom Resource

Definition (CRD) ermöglicht es Ihnen, Trident-Backends direkt über die Kubernetes-Oberfläche zu erstellen und zu verwalten. Sie können dies tun, indem Sie `kubectl` oder das entsprechende CLI-Tool für Ihre Kubernetes-Distribution verwenden.

`TridentBackendConfig`

`TridentBackendConfig` (`tbc`, `tbconfig`, `tbackendconfig`) ist ein Frontend mit Namespaces, das Ihnen ermöglicht, Trident-Backends mithilfe von `kubectl` zu verwalten. Kubernetes- und Speicheradministratoren können jetzt Backends direkt über die Kubernetes-CLI erstellen und verwalten, ohne ein spezielles Befehlszeilenprogramm zu benötigen (`tridentctl`).

Bei der Erstellung eines `TridentBackendConfig` Objekts geschieht Folgendes:

- Ein Backend wird von Trident automatisch auf Basis der von Ihnen bereitgestellten Konfiguration erstellt. Dies wird intern als `TridentBackend` (`tbe`, `tridentbackend`) CR dargestellt.
- Das `TridentBackendConfig` ist eindeutig an ein `TridentBackend` gebunden, das von Trident erstellt wurde.

Jedes `TridentBackendConfig` pflegt eine Eins-zu-Eins-Zuordnung zu einem `TridentBackend`. Ersteres ist die dem Benutzer zur Verfügung gestellte Schnittstelle zum Entwerfen und Konfigurieren von Backends; Letzteres ist die Art und Weise, wie Trident das eigentliche Backend-Objekt repräsentiert.

WARNUNG	`TridentBackend`CRs werden automatisch von Trident erstellt. Sie **sollten** diese nicht bearbeiten. Wenn Sie Aktualisierungen an Backends vornehmen möchten, tun Sie dies, indem Sie das `TridentBackendConfig` Objekt bearbeiten.

Siehe das folgende Beispiel für das 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 "trident-installer" directory für Beispielkonfigurationen für die gewünschte Speicherplattform bzw. den gewünschten Speicherdienst ansehen.

Die `spec` nimmt backendspezifische Konfigurationsparameter entgegen. In diesem Beispiel verwendet das Backend den `ontap-san` Speichertreiber und verwendet die hier tabellarisch aufgeführten

Konfigurationsparameter. Eine Liste der Konfigurationsoptionen für Ihren gewünschten Speichertreiber finden Sie in der "Backend-Konfigurationsinformationen für Ihren Speichertreiber".

Der `spec` Abschnitt umfasst außerdem `credentials` und `deletionPolicy` Felder, die neu im `TridentBackendConfig` CR eingeführt wurden:

- `credentials`: Dieser Parameter ist ein Pflichtfeld und enthält die Anmeldeinformationen, die zur Authentifizierung beim Speichersystem/-dienst verwendet werden. Dies wird auf ein vom Benutzer erstelltes Kubernetes Secret gesetzt. Die Anmeldeinformationen dürfen nicht im Klartext übergeben werden und führen sonst zu einem Fehler.
- `deletionPolicy`: Dieses Feld definiert, was geschehen soll, wenn der `TridentBackendConfig` gelöscht wird. Es kann einen von zwei möglichen Werten annehmen:
  - `delete`: Dies führt zur Löschung sowohl des `TridentBackendConfig` CR als auch des zugehörigen Backends. Dies ist der Standardwert.
  - `retain`: Wenn ein `TridentBackendConfig` CR gelöscht wird, bleibt die Backend-Definition weiterhin vorhanden und kann mit `tridentctl` verwaltet werden. Das Festlegen der Löschrichtlinie auf `retain` ermöglicht es Benutzern, auf eine frühere Version (vor 21.04) downzugraden und die erstellten Backends beizubehalten. Der Wert für dieses Feld kann aktualisiert werden, nachdem ein `TridentBackendConfig` erstellt wurde.

HINWEIS	Der Name eines Backends wird mit `spec.backendName` festgelegt. Wenn nicht angegeben, wird der Name des Backends auf den Namen des `TridentBackendConfig` Objekts (metadata.name) gesetzt. Es wird empfohlen, Backend-Namen explizit mit `spec.backendName` festzulegen.
TIPP	Backends, die mit `tridentctl` erstellt wurden, haben kein zugeordnetes `TridentBackendConfig` Objekt. Sie können solche Backends mit `kubectl` verwalten, indem Sie eine `TridentBackendConfig` CR erstellen. Achten Sie darauf, identische Konfigurationsparameter (wie `spec.backendName`, `spec.storagePrefix`, `spec.storageDriverName` usw.) anzugeben. Trident bindet die neu erstellte `TridentBackendConfig` automatisch an das bereits vorhandene Backend.

Schrittübersicht

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

1. Erstellen Sie ein "Kubernetes Secret". Das Geheimnis enthält die Anmeldeinformationen, die Trident benötigt, um mit dem Speichercluster/-dienst zu kommunizieren.
2. Erstellen Sie ein `TridentBackendConfig` Objekt. Dieses enthält spezifische Informationen zum Speicher-Cluster/-Dienst und verweist auf das im vorherigen Schritt erstellte Geheimnis.

Nachdem Sie ein Backend erstellt haben, können Sie dessen Status mithilfe von `kubectl get tbc <tbc-name> -n <trident-namespace>` beobachten und weitere Details abrufen.

Schritt 1: Erstellen Sie ein Kubernetes-Secret

Erstellen Sie ein Secret, das die Zugangsdaten für das Backend enthält. Dies ist für jeden Speicherdienst bzw. jede Plattform einzigartig. Hier ist 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
```

Diese Tabelle fasst die Felder zusammen, die im Secret für jede Speicherplattform enthalten sein müssen:

Beschreibung der Secret Fields der Storage-Plattform	Geheimnis	Feldbeschreibung
Azure NetApp Files	clientID	Die Client-ID aus einer App-Registrierung
Element (NetApp HCI/SolidFire)	Endpunkt	MVIP für den SolidFire Cluster mit Mandantenanmeldeinformationen
ONTAP	Benutzername	Benutzername für die Verbindung zum Cluster/SVM. Wird für die anmeldeinformationsbasierte Authentifizierung verwendet
ONTAP	Passwort	Passwort für die Verbindung zum Cluster/SVM. Wird für die anmeldeinformationsbasierte Authentifizierung verwendet
ONTAP	clientPrivateKey	Base64-kodierter Wert des privaten Client-Schlüssels. Wird für die zertifikatsbasierte Authentifizierung verwendet
ONTAP	chapUsername	Benutzername für eingehende Anfragen. Erforderlich, wenn useCHAP=true. Für `ontap-san` und `ontap-san-economy`
ONTAP	chapInitiatorSecret	CHAP-Initiatorgeheimnis. Erforderlich, wenn useCHAP=true. Für `ontap-san` und `ontap-san-economy`

Beschreibung der Secret Fields der Storage-Plattform	Geheimnis	Feldbeschreibung
ONTAP	chapTargetUsername	Zielbenutzername. Erforderlich, wenn useCHAP=true. Für `ontap-san` und `ontap-san-economy`
ONTAP	chapTargetInitiatorSecret	CHAP-Zielinitiatorgeheimnis. Erforderlich, wenn useCHAP=true. Für `ontap-san` und `ontap-san-economy`

Das in diesem Schritt erstellte Geheimnis wird im `spec.credentials` Feld des `TridentBackendConfig` Objekts referenziert, das im nächsten Schritt erstellt wird.

Schritt 2: Erstellen Sie die `TridentBackendConfig` CR

Sie sind nun bereit, Ihre `TridentBackendConfig` CR zu erstellen. In diesem Beispiel wird ein Backend, das den `ontap-san` Treiber verwendet, mithilfe des unten gezeigten `TridentBackendConfig` Objekts erstellt:

```
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 der `TridentBackendConfig` CR

Nachdem Sie die `TridentBackendConfig`CR erstellt haben, können Sie den Status überprüfen. Siehe das folgende Beispiel:

```
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 Backend wurde erfolgreich erstellt und an das `TridentBackendConfig` CR gebunden.

Die Phase kann einen der folgenden Werte annehmen:

- `Bound`: Der `TridentBackendConfig` CR ist mit einem Backend verknüpft, und dieses Backend enthält `configRef`, das auf die `TridentBackendConfig` UID des CR festgelegt ist.
- `Unbound`: Dargestellt durch `""`. Das `TridentBackendConfig`Objekt ist nicht an ein Backend gebunden. Alle neu erstellten `TridentBackendConfig`CRs befinden sich standardmäßig in dieser Phase. Nach dem Phasenwechsel kann es nicht wieder in den Zustand Unbound zurückversetzt werden.
- `Deleting`: Der `TridentBackendConfig` CR `deletionPolicy` war zum Löschen markiert. Wenn der `TridentBackendConfig` CR gelöscht wird, wechselt er in den Status „Wird gelöscht“.
  - Wenn auf dem Backend keine persistenten Volumenansprüche (PVCs) vorhanden sind, führt das Löschen des `TridentBackendConfig` dazu, dass Trident sowohl das Backend als auch den `TridentBackendConfig` CR löscht.
  - Sind ein oder mehrere PVCs im Backend vorhanden, wechselt es in den Löschstatus. Die `TridentBackendConfig` CR wechselt anschließend ebenfalls in die Löschphase. Das Backend und `TridentBackendConfig` werden erst gelöscht, nachdem alle PVCs gelöscht wurden.
- `Lost`: Das mit dem `TridentBackendConfig` CR verknüpfte Backend wurde versehentlich oder absichtlich gelöscht und der `TridentBackendConfig` CR enthält weiterhin eine Referenz auf das gelöschte Backend. Der `TridentBackendConfig` CR kann weiterhin gelöscht werden, unabhängig vom `deletionPolicy` Wert.
- `Unknown`: Trident kann den Status oder die Existenz des mit der `TridentBackendConfig` CR verknüpften Backends nicht feststellen. Zum Beispiel, wenn der API-Server nicht antwortet oder die `tridentbackends.trident.netapp.io` CRD fehlt. Dies könnte einen Eingriff erfordern.

In diesem Stadium ist ein Backend erfolgreich erstellt worden! Es gibt mehrere Operationen, die zusätzlich durchgeführt werden können, wie "Backend-Aktualisierungen und Backend-Löschungen".

(Optional) Schritt 4: Weitere Details einholen

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 `TridentBackendConfig` erhalten.

```
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 die `backendName` und die `backendUUID` des Backends, das als Reaktion auf die `TridentBackendConfig` CR erstellt wurde. Das `lastOperationStatus` Feld repräsentiert den Status der letzten Operation der `TridentBackendConfig` CR, die entweder vom Benutzer ausgelöst wurde (zum Beispiel, wenn der Benutzer etwas in `spec` geändert hat) oder von Trident ausgelöst wurde (zum Beispiel während Trident-Neustarts). Es kann entweder Erfolgreich oder Fehlgeschlagen sein. `phase` repräsentiert den Status der Beziehung zwischen der `TridentBackendConfig` CR und dem Backend. Im obigen Beispiel `phase` hat den Wert Bound, was bedeutet, dass die `TridentBackendConfig` CR mit dem Backend verknüpft ist.

Sie können den `kubectl -n trident describe tbc <tbc-cr-name>` Befehl ausführen, um Details zu den Ereignisprotokollen zu erhalten.

WARNUNG	Sie können ein Backend, das ein zugehöriges `TridentBackendConfig` Objekt enthält, nicht mit `tridentctl` aktualisieren oder löschen. Um die Schritte zum Wechsel zwischen `tridentctl` und `TridentBackendConfig` zu verstehen, "Siehe hier".

Backends verwalten

Führen Sie die Backend-Verwaltung mit kubectl durch

Erfahren Sie, wie Sie Backend-Verwaltungsvorgänge mithilfe von `kubectl` durchführen.

Ein Backend löschen

Durch das Löschen eines `TridentBackendConfig` weisen Sie Trident an, Backends zu löschen oder beizubehalten (basierend auf `deletionPolicy`). Um ein Backend zu löschen, stellen Sie sicher, dass `deletionPolicy` auf „delete“ gesetzt ist. Um nur das `TridentBackendConfig` zu löschen, stellen Sie sicher, dass `deletionPolicy` auf „retain“ gesetzt ist. Dadurch wird sichergestellt, dass das Backend weiterhin vorhanden ist und mit `tridentctl` verwaltet werden kann.

Führen Sie den folgenden Befehl aus:

```
kubectl delete tbc <tbc-name> -n trident
```

Trident löscht die Kubernetes-Secrets, die von `TridentBackendConfig` verwendet wurden, nicht. Der Kubernetes-Benutzer ist für die Bereinigung der Secrets verantwortlich. Beim Löschen von Secrets ist Vorsicht geboten. Sie sollten Secrets nur löschen, wenn sie von den Backends nicht verwendet werden.

Vorhandene Backends anzeigen

Führen Sie den folgenden Befehl aus:

```
kubectl get tbc -n trident
```

Sie können auch `tridentctl get backend -n trident` oder `tridentctl get backend -o yaml -n trident` ausführen, um eine Liste aller vorhandenen Backends zu erhalten. Diese Liste enthält auch Backends, die mit `tridentctl` erstellt wurden.

Aktualisieren Sie ein Backend

Es kann mehrere Gründe geben, ein Backend zu aktualisieren:

- Die Zugangsdaten für das Speichersystem haben sich geändert. Um die Zugangsdaten zu aktualisieren, muss das Kubernetes-Secret, das im `TridentBackendConfig` Objekt verwendet wird, aktualisiert werden. Trident aktualisiert das Backend automatisch mit den neuesten Zugangsdaten. Führen Sie den folgenden Befehl aus, um das Kubernetes-Secret zu aktualisieren:

  ```
  kubectl apply -f <updated-secret-file.yaml> -n trident
  ```

- Parameter (wie zum Beispiel der Name der verwendeten ONTAP SVM) müssen aktualisiert werden.
  - Sie können `TridentBackendConfig` Objekte direkt über Kubernetes mit dem folgenden Befehl aktualisieren:

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

  - Alternativ können Sie Änderungen an der bestehenden `TridentBackendConfig` CR mit dem folgenden Befehl vornehmen:

```
kubectl edit tbc <tbc-name> -n trident
```

HINWEIS

- Schlägt ein Backend-Update fehl, bleibt das Backend in seiner zuletzt bekannten Konfiguration. Sie können die Protokolle einsehen, um die Ursache zu ermitteln, indem Sie `kubectl get tbc <tbc-name> -o yaml -n trident` oder `kubectl describe tbc <tbc-name> -n trident` ausführen.
- Nachdem Sie das Problem mit der Konfigurationsdatei identifiziert und behoben haben, können Sie den Aktualisierungsbefehl erneut ausführen.

Führen Sie die Backend-Verwaltung mit tridentctl durch

Erfahren Sie, wie Sie Backend-Verwaltungsvorgänge mithilfe von `tridentctl` durchführen.

Erstelle ein Backend

Nachdem Sie eine "Konfigurationsdatei für das Backend" erstellt haben, führen Sie den folgenden Befehl aus:

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

Wenn die Backend-Erstellung fehlschlägt, gab es ein Problem mit der Backend-Konfiguration. Sie können die Protokolle einsehen, um die Ursache zu ermitteln, indem Sie den folgenden Befehl ausführen:

```
tridentctl logs -n trident
```

Nachdem Sie das Problem mit der Konfigurationsdatei identifiziert und behoben haben, können Sie einfach den `create` Befehl erneut ausführen.

Ein Backend löschen

Um ein Backend aus Trident zu löschen, gehen Sie wie folgt vor:

1. Den Backend-Namen abrufen:

```
tridentctl get backend -n trident
```

2. Backend löschen:

```
tridentctl delete backend <backend-name> -n trident
```

HINWEIS Falls Trident Volumes und Snapshots von diesem Backend bereitgestellt hat, die noch existieren, verhindert das Löschen des Backends, dass neue Volumes von ihm bereitgestellt werden. Das Backend verbleibt im Status „Wird gelöscht“.

Vorhandene Backends anzeigen

Um die von Trident bekannten Backends anzuzeigen, gehen Sie wie folgt vor:

- Um eine Zusammenfassung zu erhalten, führen Sie den folgenden Befehl aus:

```
tridentctl get backend -n trident
```

- Um alle Details zu erhalten, führen Sie den folgenden Befehl aus:

```
tridentctl get backend -o json -n trident
```

Aktualisieren Sie ein Backend

Nachdem Sie eine neue Backend-Konfigurationsdatei erstellt haben, führen Sie den folgenden Befehl aus:

```
tridentctl update backend <backend-name> -f <backend-file> -n trident
```

Wenn die Backend-Aktualisierung fehlschlägt, lag ein Fehler in der Backend-Konfiguration vor oder Sie haben eine ungültige Aktualisierung versucht. Sie können die Protokolle einsehen, um die Ursache zu ermitteln, indem Sie den folgenden Befehl ausführen:

```
tridentctl logs -n trident
```

Nachdem Sie das Problem mit der Konfigurationsdatei identifiziert und behoben haben, können Sie einfach den `update` Befehl erneut ausführen.

Identifizieren Sie die Speicherklassen, die ein Backend verwenden

Dies ist ein Beispiel für die Art von Fragen, die Sie mit dem JSON, das `tridentctl` für Backend-Objekte ausgibt, beantworten können. Dies verwendet das `jq` Hilfsprogramm, das Sie installieren müssen.

```
tridentctl get backend -o json | jq '[.items[] | {backend: .name,
storageClasses: [.storage[].storageClasses]|unique}]'
```

Dies gilt auch für Backends, die mit `TridentBackendConfig` erstellt wurden.

Wechseln Sie zwischen Backend-Verwaltungsoptionen

Erfahren Sie mehr über die verschiedenen Möglichkeiten, Backends in Trident zu verwalten.

Optionen zur Verwaltung von Backends

Mit der Einführung von `TridentBackendConfig` haben Administratoren nun zwei einzigartige Möglichkeiten, Backends zu verwalten. Dies wirft folgende Fragen auf:

- Können Backends, die mit `tridentctl` erstellt wurden, mit `TridentBackendConfig` verwaltet werden?
- Können Backends, die mit `TridentBackendConfig` erstellt wurden, mit `tridentctl` verwaltet werden?

Backends mit `tridentctl` verwalten unter Verwendung von `TridentBackendConfig`

Dieser Abschnitt beschreibt die Schritte, die erforderlich sind, um Backends zu verwalten, die mit `tridentctl` direkt über die Kubernetes-Schnittstelle durch das Erstellen von `TridentBackendConfig` Objekten erstellt wurden.

Dies gilt für die folgenden Szenarien:

- Vorhandene Backends, die kein `TridentBackendConfig` haben, weil sie mit `tridentctl` erstellt wurden.
- Neue Backends, die mit `tridentctl` erstellt wurden, während andere `TridentBackendConfig` Objekte existieren.

In beiden Szenarien bleiben die Backends weiterhin vorhanden, wobei Trident die Volumes plant und auf ihnen arbeitet. Administratoren haben hier zwei Möglichkeiten:

- Verwenden Sie `tridentctl` weiterhin, um Backends zu verwalten, die damit erstellt wurden.
- Binden Sie Backends, die mit `tridentctl` erstellt wurden, an ein neues `TridentBackendConfig` Objekt. Dadurch werden die Backends mit `kubectl` und nicht mit `tridentctl` verwaltet.

Um ein bestehendes Backend mit `kubectl` zu verwalten, müssen Sie eine `TridentBackendConfig` erstellen, die an das bestehende Backend gebunden ist. Hier ist eine Übersicht, wie das funktioniert:

1. Erstellen Sie ein Kubernetes-Secret. Das Secret enthält die Anmeldeinformationen, die Trident für die Kommunikation mit dem Speichercluster bzw. -dienst benötigt.
2. Erstellen Sie ein `TridentBackendConfig` Objekt. Dieses enthält spezifische Informationen zum Speicher-Cluster/-Dienst und verweist auf das im vorherigen Schritt erstellte Geheimnis. Achten Sie darauf, identische Konfigurationsparameter anzugeben (wie `spec.backendName`, `spec.storagePrefix`, `spec.storageDriverName` usw.). `spec.backendName` muss auf den Namen des vorhandenen Backends gesetzt werden.

Schritt 0: Backend identifizieren

Um ein `TridentBackendConfig` zu erstellen, das an ein bestehendes Backend gebunden ist, müssen Sie die Backend-Konfiguration abrufen. In diesem Beispiel nehmen wir an, dass ein Backend mit 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 1: Erstellen Sie ein Kubernetes-Secret

Erstellen Sie ein Secret, das 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 einen `TridentBackendConfig`CR

Der nächste Schritt besteht darin, eine `TridentBackendConfig` CR zu erstellen, die automatisch an die bereits vorhandene `ontap-nas-backend` gebunden wird (wie in diesem Beispiel). Stellen Sie sicher, dass die folgenden Anforderungen erfüllt sind:

- Der gleiche Backend-Name ist definiert in `spec.backendName`.
- Die Konfigurationsparameter sind identisch mit dem ursprünglichen Backend.
- Virtuelle Pools (sofern vorhanden) müssen die gleiche Reihenfolge wie im ursprünglichen Backend beibehalten.
- Anmeldeinformationen werden über ein Kubernetes Secret und nicht im Klartext bereitgestellt.

In diesem Fall wird die `TridentBackendConfig`folgendermaßen 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 der `TridentBackendConfig` CR

Nachdem das `TridentBackendConfig` erstellt wurde, muss seine Phase `Bound` sein. Es sollte außerdem denselben Backend-Namen und dieselbe UUID wie das bestehende Backend aufweisen.

```
kubectl get tbc tbc-ontap-nas-backend -n trident
NAME                   BACKEND NAME          BACKEND UUID
PHASE   STATUS
tbc-ontap-nas-backend  ontap-nas-backend     52f2eb10-e4c6-4160-99fc-
96b3be5ab5d7   Bound   Success

#confirm that no new backends were created (i.e., TridentBackendConfig did
not end up creating a new backend)
tridentctl get backend -n trident
+---------------------+----------------
+--------------------------------------+--------+---------+
|          NAME       | STORAGE DRIVER |                 UUID
| STATE  | VOLUMES |
+---------------------+----------------
+--------------------------------------+--------+---------+
| ontap-nas-backend   | ontap-nas      | 52f2eb10-e4c6-4160-99fc-
96b3be5ab5d7 | online |      25 |
+---------------------+----------------
+--------------------------------------+--------+---------+
```

Das Backend wird nun vollständig über das `tbc-ontap-nas-backend` `TridentBackendConfig` Objekt verwaltet.

Backends mit `TridentBackendConfig` verwalten unter Verwendung von `tridentctl`

```
`tridentctl` kann verwendet werden, um Backends aufzulisten, die mit
`TridentBackendConfig` erstellt wurden. Darüber hinaus können
Administratoren solche Backends auch vollständig über  `tridentctl`
verwalten, indem sie  `TridentBackendConfig` löschen und sicherstellen,
dass  `spec.deletionPolicy` auf  `retain` gesetzt ist.
```

Schritt 0: Backend identifizieren

Nehmen wir beispielsweise an, das folgende Backend wurde mit `TridentBackendConfig` erstellt:

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

Aus der Ausgabe geht hervor, dass `TridentBackendConfig` erfolgreich erstellt wurde und mit einem Backend verbunden ist [beachten Sie die UUID des Backends].

Schritt 1: Bestätigen Sie, dass `deletionPolicy` auf `retain` gesetzt ist

Betrachten wir den Wert von `deletionPolicy`. Dieser muss auf `retain`gesetzt werden. Dadurch wird sichergestellt, dass beim Löschen eines `TridentBackendConfig` CR die Backend-Definition weiterhin vorhanden ist und mit `tridentctl`verwaltet werden kann.

```
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
```

HINWEIS	Fahren Sie nicht mit dem nächsten Schritt fort, es sei denn, `deletionPolicy` ist auf `retain` gesetzt.

Schritt 2: Löschen Sie das `TridentBackendConfig` CR

Der letzte Schritt besteht darin, die `TridentBackendConfig` CR zu löschen. Nachdem Sie bestätigt haben, dass das `deletionPolicy` auf `retain` gesetzt ist, können Sie mit dem Löschen 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 |
+-------------------+----------------
+--------------------------------------+--------+---------+
```

Beim Löschen des `TridentBackendConfig` Objekts entfernt Trident dieses einfach, ohne das Backend selbst zu löschen.

Speicherklassen erstellen und verwalten

Erstellen Sie eine Speicherklasse

Konfigurieren Sie ein Kubernetes-StorageClass-Objekt und erstellen Sie die Storage-Klasse, um Trident anzuweisen, wie Volumes bereitgestellt werden.

Konfigurieren eines Kubernetes StorageClass Objekts

Das "Kubernetes StorageClass-Objekt" identifiziert Trident als den Provisioner, der für diese Klasse verwendet wird, und weist Trident an, wie ein Volume bereitgestellt werden soll. Beispiel:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-gold
provisioner: csi.trident.netapp.io
mountOptions:
  - nfsvers=3
  - nolock
parameters:
  backendType: "ontap-nas"
  media: "ssd"
allowVolumeExpansion: true
volumeBindingMode: Immediate
```

Siehe "Kubernetes- und Trident-Objekte" für Details dazu, wie Storage-Klassen mit dem `PersistentVolumeClaim` interagieren und Parameter zur Steuerung, wie Trident Volumes bereitstellt.

Erstellen Sie eine Speicherklasse

Nachdem Sie das StorageClass-Objekt erstellt haben, können Sie die Speicherklasse erstellen. Storage-Class-Beispiele bietet einige grundlegende Beispiele, die Sie verwenden oder anpassen können.

Schritte

1. Dies ist ein Kubernetes-Objekt, daher verwenden Sie `kubectl`, um es in Kubernetes zu erstellen.

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

2. Sie sollten nun eine **basic-csi** Storage-Class sowohl in Kubernetes als auch in Trident sehen, und Trident sollte die Pools im Backend erkannt haben.

```
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"
        ]
      }
    }
  ]
}
```

Storage-Class-Beispiele

Trident bietet "einfache Speicherklassendefinitionen für spezifische Backends".

Alternativ können Sie die `sample-input/storage-class-csi.yaml.templ` Datei, die mit dem Installationsprogramm geliefert wird, bearbeiten und *`BACKEND_TYPE`* durch den Namen des Speichertreibers ersetzen.

```
./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
```

Speicherklassen verwalten

Sie können vorhandene Speicherklassen anzeigen, eine Standard-Speicherklasse festlegen, die Speicherklassen-Backends identifizieren und Speicherklassen löschen.

Vorhandene Speicherklassen anzeigen

- Um vorhandene Kubernetes-Speicherklassen anzuzeigen, führen Sie den folgenden Befehl aus:

```
kubectl get storageclass
```

- Um Details zur Kubernetes-Speicherklasse anzuzeigen, führen Sie den folgenden Befehl aus:

```
kubectl get storageclass <storage-class> -o json
```

- Um die synchronisierten Speicherklassen von Trident anzuzeigen, führen Sie den folgenden Befehl aus:

```
tridentctl get storageclass
```

- Um die Details der synchronisierten Speicherklasse von Trident anzuzeigen, führen Sie den folgenden Befehl aus:

```
tridentctl get storageclass <storage-class> -o json
```

Legen Sie eine Standardspeicherklasse fest

Kubernetes 1.6 führte die Möglichkeit ein, eine Standard-Speicherklasse festzulegen. Dies ist die Speicherklasse, die für die Bereitstellung eines Persistent Volume verwendet wird, wenn ein Benutzer in einem Persistent Volume Claim (PVC) keine angibt.

- Definieren Sie eine Standard-Speicherklasse, indem Sie die Annotation `storageclass.kubernetes.io/is-default-class` auf true in der Speicherklassendefinition setzen. Gemäß der Spezifikation wird jeder andere Wert oder das Fehlen der Annotation als false interpretiert.
- Sie können eine vorhandene Speicherklasse als Standardspeicherklasse konfigurieren, indem Sie den folgenden Befehl verwenden:

```
kubectl patch storageclass <storage-class-name> -p '{"metadata":
{"annotations":{"storageclass.kubernetes.io/is-default-class":"true"}}}'
```

- Ebenso können Sie die Standard-Speicherklassenannotation mit dem folgenden Befehl entfernen:

```
kubectl patch storageclass <storage-class-name> -p '{"metadata":
{"annotations":{"storageclass.kubernetes.io/is-default-class":"false"}}}'
```

Im Trident-Installationspaket sind auch Beispiele enthalten, die diese Annotation enthalten.

HINWEIS	Es sollte immer nur eine Standard-Speicherklasse in Ihrem Cluster vorhanden sein. Kubernetes verhindert dies technisch zwar nicht, aber es verhält sich so, als gäbe es überhaupt keine Standard-Speicherklasse.

Identifizieren Sie das Backend für eine Storage Class

Dies ist ein Beispiel für die Art von Fragen, die Sie mit dem JSON, das `tridentctl` für Trident Backend-Objekte ausgibt, beantworten können. Dies verwendet das `jq` Hilfsprogramm, das Sie möglicherweise zuerst installieren müssen.

```
tridentctl get storageclass -o json | jq  '[.items[] | {storageClass:
.Config.name, backends: [.storage]|unique}]'
```

Eine Storage Class löschen

Um eine Speicherklasse aus Kubernetes zu löschen, führen Sie den folgenden Befehl aus:

```
kubectl delete storageclass <storage-class>
```

`<storage-class>` sollte durch Ihre Speicherklasse ersetzt werden.

Alle persistenten Volumes, die über diese Speicherklasse erstellt wurden, bleiben unberührt, und Trident wird sie weiterhin verwalten.

HINWEIS	Trident erzwingt ein `fsType` für die von ihm erstellten Volumes. Für iSCSI-Backends wird empfohlen, `parameters.fsType` in der StorageClass zu erzwingen. Sie sollten vorhandene StorageClasses löschen und sie mit `parameters.fsType` neu erstellen.

Volumes bereitstellen und verwalten

Ein Volume bereitstellen

Erstellen Sie einen PersistentVolumeClaim (PVC), der die konfigurierte Kubernetes-StorageClass verwendet, um Zugriff auf das PV anzufordern. Anschließend können Sie das PV in einen Pod einbinden.

Überblick

Ein "*PersistentVolumeClaim*" (PVC) ist eine Anfrage für den Zugriff auf das PersistentVolume im Cluster.

Die PVC kann so konfiguriert werden, dass sie Speicherplatz einer bestimmten Größe oder eines bestimmten Zugriffsmodus anfordert. Mithilfe der zugehörigen StorageClass kann der Clusteradministrator mehr als nur PersistentVolume-Größe und Zugriffsmodus steuern – zum Beispiel Leistung oder Servicelevel.

Nachdem Sie das PVC erstellt haben, können Sie das Volume in einem Pod einbinden.

PVC erstellen

Schritte

1. Erstellen Sie das PVC.

```
kubectl create -f pvc.yaml
```

2. Überprüfen Sie den PVC-Status.

```
kubectl get pvc
```

```
NAME          STATUS   VOLUME     CAPACITY   ACCESS MODES   STORAGECLASS AGE
pvc-storage   Bound    pv-name    1Gi        RWO                    5m
```

1. Mountieren Sie das Volume in einem Pod.

```
kubectl create -f pv-pod.yaml
```

HINWEIS	Sie können den Fortschritt mit `kubectl get pod --watch` überwachen.

2. Überprüfen Sie, ob das Volume auf `/my/mount/path` eingebunden ist.

```
kubectl exec -it task-pv-pod -- df -h /my/mount/path
```

3. Sie können den Pod nun löschen. Die Pod-Anwendung existiert dann nicht mehr, aber das Volume bleibt erhalten.

```
kubectl delete pod pv-pod
```

Beispielmanifeste

PersistentVolumeClaim-Beispielmanifeste

Diese Beispiele zeigen grundlegende PVC-Konfigurationsoptionen.

PVC mit RWO-Zugriff

Dieses Beispiel zeigt eine einfache PVC mit RWO-Zugriff, die mit einer StorageClass namens `basic-csi` verknüpft ist.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc-storage
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: basic-csi
```

PVC mit NVMe/TCP

Dieses Beispiel zeigt eine einfache PVC für NVMe/TCP mit RWO-Zugriff, die einer StorageClass namens `protection-gold` zugeordnet ist.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
name: pvc-san-nvme
spec:
accessModes:
  - ReadWriteOnce
resources:
  requests:
    storage: 300Mi
storageClassName: protection-gold
```

Pod-Manifestbeispiele

Diese Beispiele zeigen grundlegende Konfigurationen, um das PVC an ein Pod anzuhängen.

Grundkonfiguration

```
kind: Pod
apiVersion: v1
metadata:
  name: pv-pod
spec:
  volumes:
    - name: storage
      persistentVolumeClaim:
       claimName: pvc-storage
  containers:
    - name: pv-container
      image: nginx
      ports:
        - containerPort: 80
          name: "http-server"
      volumeMounts:
        - mountPath: "/my/mount/path"
          name: storage
```

Grundlegende NVMe/TCP-Konfiguration

```
apiVersion: v1
kind: Pod
metadata:
  name: pod-nginx
spec:
  volumes:
    - name: basic-pvc
      persistentVolumeClaim:
        claimName: pvc-san-nvme
  containers:
    - name: task-pv-container
      image: nginx
      volumeMounts:
        - mountPath: "/my/mount/path"
          name: basic-pvc
```

Siehe "Kubernetes- und Trident-Objekte" für Details dazu, wie Storage-Klassen mit dem `PersistentVolumeClaim` interagieren und Parameter zur Steuerung, wie Trident Volumes bereitstellt.

Volumen erweitern

Trident ermöglicht Kubernetes-Nutzern die nachträgliche Erweiterung ihrer Volumes. Finden Sie Informationen zu den erforderlichen Konfigurationen, um iSCSI-, NFS-, SMB-, NVMe/TCP- und FC-Volumes zu erweitern.

Ein iSCSI-Volume erweitern

Sie können ein iSCSI Persistent Volume (PV) mithilfe des CSI provisioner erweitern.

HINWEIS	Die iSCSI-Volume-Erweiterung wird von den `ontap-san`, `ontap-san-economy`, `solidfire-san` Treibern unterstützt und erfordert Kubernetes 1.16 oder höher.

Schritt 1: Konfigurieren Sie die StorageClass, um die Volumenerweiterung zu unterstützen

Bearbeiten Sie die StorageClass-Definition, um das `allowVolumeExpansion` Feld auf `true` zu setzen.

```
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 eine bereits bestehende StorageClass, um den `allowVolumeExpansion` Parameter hinzuzufügen.

Schritt 2: Erstellen Sie eine PVC mit der StorageClass, die Sie erstellt haben

Bearbeiten Sie die PVC-Definition und aktualisieren Sie die `spec.resources.requests.storage`, um die neu gewünschte Größe anzugeben, die größer als die ursprüngliche Größe sein muss.

```
cat pvc-ontapsan.yaml
```

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: san-pvc
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: ontap-san
```

Trident erstellt ein Persistent Volume (PV) und ordnet es diesem Persistent Volume Claim (PVC) zu.

```
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 ein Pod, das das PVC anbindet

Befestigen Sie das PV an einem Pod, damit es in der Größe angepasst werden kann. Es gibt zwei Szenarien, wenn die Größe eines iSCSI PV geändert wird:

- Wenn das PV an einen Pod angehängt ist, erweitert Trident das Volume auf dem Storage-Backend, scannt das Gerät erneut und passt die Größe des Dateisystems an.
- Beim Versuch, ein nicht zugeordnetes PV zu vergrößern, erweitert Trident das Volume im Storage-Backend. Nachdem das PVC an einen Pod gebunden wurde, scannt Trident das Gerät erneut und passt die Größe des Dateisystems an. Kubernetes aktualisiert dann die PVC-Größe, nachdem der Erweiterungsvorgang erfolgreich abgeschlossen wurde.

In diesem Beispiel wird ein Pod erstellt, der die `san-pvc` verwendet.

```
 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: PV erweitern

Um die Größe des erstellten PV von 1Gi auf 2Gi zu ändern, bearbeiten Sie die PVC-Definition und aktualisieren Sie die `spec.resources.requests.storage` auf 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 überprüfen, ob die Erweiterung korrekt funktioniert hat, indem Sie die Größe des PVC, PV und des Trident volume kontrollieren:

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

Ein FC-Volume erweitern

Sie können ein FC Persistent Volume (PV) mithilfe des CSI provisioner erweitern.

HINWEIS FC-Volume-Erweiterung wird vom `ontap-san` Treiber unterstützt und erfordert Kubernetes 1.16 und höher.

Schritt 1: Konfigurieren Sie die StorageClass, um die Volumenerweiterung zu unterstützen

Bearbeiten Sie die StorageClass-Definition, um das `allowVolumeExpansion` Feld auf `true` zu setzen.

```
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 eine bereits bestehende StorageClass, um den `allowVolumeExpansion` Parameter hinzuzufügen.

Schritt 2: Erstellen Sie eine PVC mit der StorageClass, die Sie erstellt haben

Bearbeiten Sie die PVC-Definition und aktualisieren Sie die `spec.resources.requests.storage`, um die neu gewünschte Größe anzugeben, die größer als die ursprüngliche Größe sein muss.

```
cat pvc-ontapsan.yaml
```

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: san-pvc
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: ontap-san
```

Trident erstellt ein Persistent Volume (PV) und ordnet es diesem Persistent Volume Claim (PVC) zu.

```
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 ein Pod, das das PVC anbindet

Befestigen Sie das PV an einem Pod, damit es in der Größe angepasst werden kann. Es gibt zwei Szenarien, wenn ein FC-PV in der Größe geändert wird:

- Wenn das PV an einen Pod angehängt ist, erweitert Trident das Volume auf dem Storage-Backend, scannt das Gerät erneut und passt die Größe des Dateisystems an.
- Beim Versuch, ein nicht zugeordnetes PV zu vergrößern, erweitert Trident das Volume im Storage-Backend. Nachdem das PVC an einen Pod gebunden wurde, scannt Trident das Gerät erneut und passt

die Größe des Dateisystems an. Kubernetes aktualisiert dann die PVC-Größe, nachdem der Erweiterungsvorgang erfolgreich abgeschlossen wurde.

In diesem Beispiel wird ein Pod erstellt, der die `san-pvc` verwendet.

```
 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: PV erweitern

Um die Größe des erstellten PV von 1Gi auf 2Gi zu ändern, bearbeiten Sie die PVC-Definition und aktualisieren Sie die `spec.resources.requests.storage` auf 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 überprüfen, ob die Erweiterung korrekt funktioniert hat, indem Sie die Größe des PVC, PV und des Trident volume kontrollieren:

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

Ein NFS-Volume erweitern

Trident unterstützt die Volume-Erweiterung für NFS PVs, die auf `ontap-nas`, `ontap-nas-economy`, `ontap-nas-flexgroup` und `azure-netapp-files` Backends bereitgestellt werden.

Schritt 1: Konfigurieren Sie die StorageClass, um die Volumenerweiterung zu unterstützen

Um die Größe eines NFS-PV zu ändern, muss der Administrator zunächst die Speicherklasse so konfigurieren, dass eine Volumenerweiterung möglich ist, indem er das `allowVolumeExpansion` Feld auf `true` setzt:

```
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
```

Falls Sie bereits eine Speicherklasse ohne diese Option erstellt haben, können Sie die bestehende

Speicherklasse einfach bearbeiten, indem Sie `kubectl edit storageclass` verwenden, um die Volumenerweiterung zu ermöglichen.

Schritt 2: Erstellen Sie eine PVC mit der StorageClass, die Sie erstellt haben

```
cat pvc-ontapnas.yaml
```

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: ontapnas20mb
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 20Mi
  storageClassName: ontapnas
```

Trident sollte eine 20 MiB 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: PV erweitern

Um die neu erstellte 20 MiB PV auf 1 GiB zu skalieren, bearbeiten Sie die PVC und setzen Sie `spec.resources.requests.storage` auf 1 GiB:

```
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 validieren, ob die Größenänderung korrekt funktioniert hat, indem Sie die Größe des PVC, PV und des Trident-Volumes 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    |
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
```

Grenzen des RWX NVMe-Subsystems verstehen

ReadWriteMany (RWX)-Volumes, die das NVMe-Protokoll verwenden, haben eine Skalierbarkeitsgrenze von 64 Knoten pro Volume. Im Folgenden werden die Einschränkungen erläutert, die Architektur des beteiligten NVMe-Subsystems erklärt und die erforderlichen Schritte zur Behebung aufgezeigt.

Verstehen Sie das 64-Knoten-Limit

Wenn Sie ReadWriteMany (RWX)-Volumes mit dem NVMe-Protokoll verwenden möchten, kann ein einzelnes RWX NVMe-Volume nicht von mehr als 64 Knoten in einem Kubernetes-Cluster eingebunden werden.

Planen Sie keine Workloads ein, die dieselbe RWX NVMe PersistentVolumeClaim auf mehr als 64 Knoten einbinden.

Diese Einschränkung gilt nur für RWX-Volumes, die das NVMe-Protokoll verwenden.

NVMe-Subsystemmodelle verstehen

Subsystemmodell pro Volumen (Trident Releases früher als 26.02)

In Trident-Versionen vor 26.02 werden RWX-NVMe-Volumes mithilfe eines Subsystemmodells pro Volume bereitgestellt. Jedes RWX-NVMe-Volume ist seinem eigenen dedizierten NVMe-Subsystem auf ONTAP zugeordnet.

Dieses Modell ist einfach, hat aber eine geringere Skalierbarkeitsgrenze. In großen Kubernetes-Clustern werden die Grenzen der Subsystem-Controller schnell erreicht, da jedes RWX-Volume ein eigenes Subsystem belegt.

Super-Subsystem-Modell (eingeführt in Trident 26.02)

Ab Trident 26.02 verwenden RWX NVMe-Volumes ein gemeinsames Super-Subsystem-Modell. Mehrere RWX NVMe-Volumes teilen sich dasselbe NVMe-Subsystem.

Jedes Super-Subsystem unterstützt bis zu 1024 Namensräume (Volumes). Dieses Modell verbessert die Skalierbarkeit für RWX-Workloads erheblich und verringert die Wahrscheinlichkeit, die ONTAP-Subsystem-Grenzen zu erreichen.

Jedes RWX NVMe-Volume unterstützt bis zu 64 Knoten.

Fehlersymptome identifizieren

Wenn Sie RWX NVMe-Volumes in großem Umfang erstellen oder einbinden, können Fehler wie die folgenden auftreten:

```
Maximum number of controllers reached. No more controllers can be created.
```

Dieser Fehler weist darauf hin, dass das ONTAP NVMe-Subsystem-Controller-Limit erreicht wurde.

Beheben Sie Subsystem-Limit-Fehler

Um die Beschränkungen des Subsystems pro Volume zu überwinden und das Super-Subsystem-Modell nutzen zu können, aktualisieren Sie auf Trident 26.02 oder höher.

Aktualisieren Sie Trident, um das Super-Subsystem-Modell anzuwenden

Um das Super-Subsystem-Modell für RWX NVMe-Volumes anzuwenden:

1. Aktualisieren Sie Trident auf Version 26.02 oder höher.
2. Skalieren Sie alle Pods, die RWX NVMe-Volumes verwenden, auf null Replikate herunter.
3. Stellen Sie sicher, dass keine Workloads aktiv RWX NVMe-Volumes verwenden.
4. Skalieren Sie die Pods wieder hoch.

Diese Neustartsequenz stellt sicher, dass RWX NVMe volumes mithilfe des Super-Subsystem-Modells eingebunden werden.

- Diese Einschränkung gilt nur für RWX-Volumes, die das NVMe-Protokoll verwenden.
- Die Beschränkung auf 64 Knoten gilt pro RWX NVMe volume.
- Andere Zugriffsmodi und andere Protokolle sind nicht betroffen.

Controller-Skalierbarkeit

Trident führt Controller-Skalierbarkeit durch verbesserte Parallelität über mehrere Speichertreiber hinweg ein. Kunden können erkennen, welche Trident-Treiber die Controller-Skalierbarkeit bei allgemeiner Verfügbarkeit unterstützen und welche Treiber als technische Vorschau in Trident 26.02 verfügbar sind. Dies ermöglicht fundierte Bereitstellungsentscheidungen und ein angemessenes Risikomanagement für skalierbare Kubernetes-Umgebungen.

Schlüsselbegriffe und Definitionen

Controller-Skalierbarkeit

Controller-Skalierbarkeit bezieht sich auf die Fähigkeit des Trident Controllers, mehrere Speicheroperationen parallel zu verarbeiten, anstatt sie hinter einer einzigen Sperre zu serialisieren. Zu diesen Operationen gehören das Erstellen, Löschen, Ändern der Größe, Erstellen und Löschen von Snapshots, das Veröffentlichen und Aufheben der Veröffentlichung von Volumes sowie die Backend-Verwaltung.

Wenn die Controller Skalierbarkeit aktiviert ist, werden Operationen auf verschiedenen Volumes und Backends parallel ausgeführt. Dies erhöht den Durchsatz und verkürzt die Gesamtbetriebszeit in Umgebungen mit einer hohen Anzahl gleichzeitiger PersistentVolumeClaim und VolumeSnapshot Operationen.

Unterstützung der Controller-Skalierbarkeit

Trident unterstützt die Skalierbarkeit des Controllers mit unterschiedlichen Reifegraden, abhängig vom Speichertreiber.

Allgemeine Verfügbarkeit

Die folgenden Treiber unterstützen die Controller-Skalierbarkeit bei allgemeiner Verfügbarkeit in Trident 26.02:

- `ontap-san`
- `ontap-nas`
- `google-cloud-netapp-volumes`

HINWEIS	Die `google-cloud-netapp-volumes` und `google-cloud-netapp-volumes-san` Treiber sind unterschiedlich. Nur `google-cloud-netapp-volumes` wird unterstützt. Verwenden Sie `google-cloud-netapp-volumes-san` nicht in Backend-Konfigurationen oder Beispielen.

Controller-Skalierbarkeit aktivieren

Controller-Skalierbarkeit wird durch die `enableConcurrency` Konfigurationsoption gesteuert. Diese Option muss während der Trident-Installation oder durch Aktualisierung einer bestehenden Bereitstellung explizit aktiviert werden.

Trident Operator-Bereitstellung

Um die Controller-Skalierbarkeit mit dem Trident Operator zu aktivieren, setzen Sie `enableConcurrency` auf `true` in der `TridentOrchestrator` benutzerdefinierten Ressource.

Neuinstallation

Erstellen oder bearbeiten Sie die `TridentOrchestrator` CR mit `enableConcurrency` auf `true` gesetzt:

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  namespace: trident
  enableConcurrency: true
```

Wenden Sie die CR an:

```
kubectl apply -f tridentorchestrator_cr.yaml
```

Vorhandene Installation

Patchen Sie die bestehende `TridentOrchestrator`CR, um die Skalierbarkeit des Controllers zu ermöglichen:

```
kubectl patch torc trident --type=merge -p
'{"spec":{"enableConcurrency":true}}'
```

Überprüfen Sie, ob die Einstellung angewendet wurde:

```
kubectl get torc trident -o
jsonpath='{.status.currentInstallationParams.enableConcurrency}'
```

Helm-Bereitstellung

Um die Controller-Skalierbarkeit mit Helm zu aktivieren, setzen Sie den `enableConcurrency` Wert auf `true`.

Neuinstallation

```
helm install trident netapp-trident/trident-operator --namespace trident
--create-namespace --set enableConcurrency=true
```

Vorhandene Installation

```
helm upgrade trident netapp-trident/trident-operator --namespace trident
--set enableConcurrency=true
```

Alternativ setzen Sie `enableConcurrency` auf `true` in einer benutzerdefinierten `values.yaml` Datei:

```
# values.yaml
enableConcurrency: true
```

Installieren oder aktualisieren Sie anschließend mithilfe der values-Datei:

```
helm install trident netapp-trident/trident-operator --namespace trident
--create-namespace -f values.yaml
```

tridentctl-Bereitstellung

Um die Skalierbarkeit des Controllers mit `tridentctl` zu ermöglichen, übergeben Sie das `--enable -concurrency` Flag während der Installation.

Neuinstallation

```
tridentctl install -n trident --enable-concurrency
```

Vorhandene Installation

Um die Skalierbarkeit des Controllers in einer bestehenden `tridentctl`-basierten Bereitstellung zu aktivieren, deinstallieren und installieren Sie mit dem Flag neu:

```
tridentctl uninstall -n trident
tridentctl install -n trident --enable-concurrency
```

Überprüfen Sie, ob die Controller-Skalierbarkeit aktiviert ist.

Nachdem Sie die Controller-Skalierbarkeit aktiviert haben, überprüfen Sie anhand der Controller-Pod-Logs, ob der Trident-Controller mit aktivierter Parallelverarbeitung ausgeführt wird:

```
kubectl logs -n trident deploy/trident-controller | grep -i concurrency
```

Sie sollten einen Logeintrag sehen, der anzeigt, dass die Parallelität aktiviert ist.

Technische Vorschau

Die folgenden Treiber unterstützen die Controller-Skalierbarkeit als technische Vorschau in Trident 26.02:

- `nas-eco`
- `san-eco`

Für diese Treiber:

- Controller-Konkurrenz ist für Evaluierung und Tests verfügbar
- Das Verhalten kann sich in zukünftigen Versionen ändern
- Die Verwendung in Produktionsumgebungen wird nicht empfohlen

Parallelitätsverhalten

Wenn die Controller-Skalierbarkeit aktiviert ist:

- Trident ersetzt die einzelne globale Sperre durch eine feingranulare, ressourcenbezogene Sperre
- Operationen, die dieselbe Ressource verändern, werden serialisiert, um die Datenkonsistenz aufrechtzuerhalten
- Operationen, die nur von einer Ressource lesen, können gleichzeitig mit anderen Leseoperationen auf dieser Ressource ausgeführt werden
- Trident begrenzt gleichzeitige ONTAP API-Anfragen auf 20 pro Management-LIF, um eine Überlastung der Backend-Speichersysteme zu verhindern
- Wenn mehrere Backends dieselbe Management-LIF verwenden, teilen sie sich dieses Limit von 20 Anfragen

Bekannte Einschränkungen und Überlegungen

Die folgenden Überlegungen gelten für die Skalierbarkeit von Controllern:

- Die Parallelverarbeitung wird intern vom Trident Controller verwaltet.
- In dieser Version gibt es keine vom Benutzer konfigurierbaren Parallelitätsbeschränkungen
- Der Gesamtdurchsatz hängt ab von:
  - Der verwendete Speichertreiber
  - Reaktionsfähigkeit
  - Kubernetes API server-Performance
- Hohe Parallelität kann die Last auf Backend-Speichersystemen erhöhen

Vorbehalte und Einschränkungen

Für Trident 26.02 gelten die folgenden Einschränkungen:

- Das Skalierbarkeitsverhalten des Controllers ist nicht bei allen Treibern identisch
- Technische Vorschau-Treiber können Folgendes aufweisen:
  - Unbeständige Leistung unter hoher Last
  - Verhaltensänderungen zwischen den Releases
- Das Debuggen von gleichzeitigen Operationen kann aufgrund der parallelen Ausführung komplexer sein
- Metriken und Protokolle können verschachtelte Betriebsausgaben anzeigen

Empfehlungen

- Verwenden Sie allgemein verfügbare (GA) Treiber für Produktionsumgebungen, die eine hohe Skalierbarkeit erfordern
- Evaluieren Sie Technical Preview-Treiber in Nicht-Produktionsumgebungen

- Überwachen Sie die Backend- und Controller-Performance beim Betrieb in großem Maßstab
- Vermeiden Sie es, die Reihenfolge von Operationen in Automatisierungsskripten anzunehmen

Automatische Volumenerweiterung

Die automatische Volumenerweiterung ermöglicht es, dass von Trident bereitgestellte Persistent Volumes automatisch wachsen, sobald die genutzte Kapazität einen definierten Schwellenwert erreicht. Diese Funktion reduziert den Betriebsaufwand und trägt dazu bei, Anwendungsunterbrechungen aufgrund von Kapazitätsengpässen zu vermeiden. Die automatische Volumenerweiterung wird mithilfe von Autogrow Policies implementiert. Eine Autogrow Policy definiert:

- Die Auslastungsschwelle, die die Erweiterung auslöst
- Die Menge, um die das Volumen wächst
- Die maximale Größe, die das Volume erreichen kann

HINWEIS	Volumes erhöhen sich automatisch, wenn der festgelegte Auslastungsschwellenwert überschritten wird. Volumes werden niemals automatisch reduziert.

Anforderungen

Bevor Sie die automatische Volumenerweiterung konfigurieren, stellen Sie sicher, dass die folgenden Voraussetzungen erfüllt sind:

- Trident 26.02 oder später
- Rollenbasierte Zugriffssteuerungsberechtigungen zum Erstellen `TridentAutogrowPolicy` benutzerdefinierter Ressourcen
- StorageClasses konfiguriert mit `allowVolumeExpansion: true`
- Unterstützte ONTAP-Protokolle:
  - Netzwerkdateisystem (NFS)
  - Internet Small Computer Systems Interface (iSCSI)
  - Non-Volatile Memory Express (NVMe)
  - Fibre Channel Protocol (FCP)

Einschränkungen

- ONTAP Non-Volatile Memory Express Raw-Block-Volumes früher als ONTAP 9.16.1 unterstützen keine automatische Erweiterung.
- Für SAN-Volumes, wenn `growthAmount` kleiner oder gleich 50 Mebibyte ist, erhöht Trident den Wert automatisch auf 51 Mebibyte, bevor die Größe angepasst wird, vorausgesetzt, die resultierende Größe überschreitet nicht `maxSize`.
- In Brownfield-Umgebungen funktioniert die automatische Erweiterung möglicherweise nicht für bestimmte vorhandene Volumes aufgrund des Migrationsverhaltens bei der Volume-Veröffentlichung.
- Wenn ein Volumen `maxSize` erreicht, findet keine weitere Erweiterung statt.
- Unterstützte Protokolle für die automatische Volumenerweiterung:

- Netzwerkdateisystem (NFS)
- Internet Small Computer Systems Interface (iSCSI)
- Non-Volatile Memory Express (NVMe)
- Fibre Channel Protocol (FCP)

Volumes mit Autogrow-Richtlinie bereitstellen

Die Autogrow Policy kann auf zwei Ebenen konfiguriert werden:

- Speicherklassenebene: Legt den Standardwert für alle Volumes fest (mittels Annotation)
- PVC-Ebene: Überschreibt die Standardeinstellung der Speicherklasse (using annotation)

Erstellen Sie eine Autogrow-Richtlinie

Autogrow-Richtlinien ermöglichen die automatische Volumenerweiterung, wenn Volumes einen definierten Kapazitätsschwellenwert erreichen.

Stellen Sie sicher, dass Sie Folgendes haben:

- Trident 26.02 oder höher installiert
- Rollenbasierte Zugriffssteuerungsberechtigungen zum Erstellen `TridentAutogrowPolicy` von Ressourcen
- Verständnis der Anforderungen an das Arbeitslastwachstum

Eine Autogrow Policy definiert, wie sich Volumes automatisch vergrößern, wenn sie einen festgelegten Kapazitätsschwellenwert erreichen.

Sie können Autogrow Policies an jeder beliebigen Stelle in Ihrem Workflow erstellen:

- Bevor StorageClasses und Volumes erstellt werden
- Nachdem StorageClasses existieren
- Nachdem die Volumes bereitgestellt wurden

Diese Flexibilität ermöglicht es Ihnen, eine automatische Erweiterung einzuführen, ohne bestehende Ressourcen neu zu erstellen.

Spezifikationen der Autogrow Policy

Autogrow Policies sind benutzerdefinierte Kubernetes-Ressourcen, die wie folgt definiert sind:

Feld	Beschreibung	Format	Erforderlich	Beispiel	Standard
Name	Eindeutige Richtlinienkennung	Zeichenkette	Ja	Produktionsdatenbankrichtlinie	Keine
usedThreshold	Kapazitätsprozentsatz, der die Erweiterung auslöst	Prozentzeichenkette	Ja	"80%"	Keine

Feld	Beschreibung	Format	Erforderlich	Beispiel	Standard
growthAmount	Betrag, um den erhöht wird, wenn der Schwellenwert erreicht ist	Prozentualer Anteil oder Größe	Nein	"10%" oder "5Gi"	"10%"
maxSize	Maximale Volumengrößenbeschränkung	Kubernetes-Menge	Nein	"500Gi"	Unbegrenzt

Erstellen Sie eine Autogrow-Richtlinie

Schritte

1. Erstellen Sie eine YAML-Datei, die Ihre Autogrow Policy definiert:

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: standard-autogrow
spec:
  usedThreshold: "80%"
  growthAmount: "10%"
  maxSize: "500Gi"
```

2. Wenden Sie die Richtlinie auf Ihren Cluster an:

```
kubectl apply -f autogrow-policy.yaml
```

3. Überprüfen Sie, ob die Richtlinie erstellt wurde:

```
kubectl get tridentautogrowpolicy standard-autogrow
```

Erwartete Ausgabe

```
NAME                USED THRESHOLD   GROWTH AMOUNT   STATE
standard-autogrow   80%              10%             Success
```

Richtlinienstatus

Nachdem Sie eine Richtlinie erstellt haben, validiert Trident die Spezifikation und weist einen der folgenden Zustände zu:

Status	Beschreibung	Aktion erforderlich
Erfolg	Die Richtlinie ist validiert und einsatzbereit.	Keine.

Status	Beschreibung	Aktion erforderlich
Fehlgeschlagen	Validierungsfehler festgestellt.	Spezifikation prüfen und korrigieren.
Löschen	Löschung wird durchgeführt.	Warten Sie auf den Abschluss.

Ordnen Sie eine Richtlinie einer StorageClass zu

Sie können eine Autogrow-Richtlinie mit einer StorageClass durch die `trident.netapp.io/autogrowPolicy` Annotation verknüpfen. Alle Volumes, die aus dieser StorageClass bereitgestellt werden, erben die Richtlinie.

HINWEIS Die StorageClass muss `allowVolumeExpansion: true` haben.

Schritte

1. Erstellen oder ändern Sie eine StorageClass mit der Autogrow Policy-Annotation:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-gold
  annotations:
    trident.netapp.io/autogrowPolicy: "production-db-policy"
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
  fsType: "ext4"
allowVolumeExpansion: true
```

2. Wenden Sie die StorageClass an:

```
kubectl apply -f storageclass.yaml
```

3. Überprüfen Sie die Annotation:

```
kubectl get storageclass ontap-gold -o
jsonpath='{.metadata.annotations.trident\.netapp\.io/autogrowPolicy}'
```

Erwartete Ausgabe

```
production-db-policy
```

Richtlinienvorrang

Wenn Autogrow-Richtlinien-Annotationen sowohl für eine StorageClass als auch für ein PVC festgelegt sind, wendet Trident die folgenden Vorrangregeln an:

1. **PVC-Annotationen haben Vorrang.** Wenn eine PVC `trident.netapp.io/autogrowPolicy` festlegt, wird dieser Wert immer verwendet.
2. **StorageClass-Anmerkung gilt nur, wenn die PVC keine Anmerkung enthält.**
3. **Wenn keines der beiden die Annotation hat,** wird keine Autogrow Policy angewendet.

StorageClass Anmerkung	PVC-Anmerkung	Effektives Verhalten
`trident.netapp.io/autogrowPolicy: standard-agp`	Nicht festgelegt	Anwendungsgebiete `standard-agp`.
`trident.netapp.io/autogrowPolicy: standard-agp`	`trident.netapp.io/autogrowPolicy: logs-policy`	Verwendet `logs-policy` (PVC überschreibt StorageClass).
`trident.netapp.io/autogrowPolicy: standard-agp`	`trident.netapp.io/autogrowPolicy: "none"`	Keine Autogrow-Richtlinie (PVC deaktiviert autogrow).
Nicht festgelegt	`trident.netapp.io/autogrowPolicy: dev-policy`	Anwendungsgebiete `dev-policy`.
Nicht festgelegt	Nicht festgelegt	Keine Autogrow Policy.

Konfigurationsbeispiele

Die folgenden Beispiele zeigen gängige Autogrow Policy-Konfigurationen für verschiedene Anwendungsfälle.

Konservative Richtlinie für Produktionsdatenbanken

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: production-db-policy
spec:
  usedThreshold: "75%"
  growthAmount: "20%"
  maxSize: "5Ti"
```

Protokollspeicherung mit festen Wachstumsschritten

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: log-storage-policy
spec:
  usedThreshold: "90%"
  growthAmount: "10Gi"
  maxSize: "100Gi"
```

Minimalrichtlinie mit Standardeinstellungen

Wenn Sie `growthAmount` und `maxSize` weglassen, verwendet Trident die Standardeinstellungen (`10%` Wachstum, unbegrenzte Größe):

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: logs-policy
spec:
  usedThreshold: "85%"
```

Richtlinie mit benutzerdefiniertem maxSize und standardmäßigem growthAmount

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: default-ga-policy
spec:
  usedThreshold: "70%"
  maxSize: "100Gi"
```

Aggressives Wachstum mit unbegrenztem maxSize

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: aggressive-growth-policy
spec:
  usedThreshold: "80%"
  growthAmount: "150%"
```

Richtlinie mit Bruchteilprozenten

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: precise-policy
spec:
  usedThreshold: "80.28%"
  growthAmount: "10.65%"
  maxSize: "100Gi"
```

HINWEIS Bruchprozentsätze werden unterstützt. Wenn Sie mehr als drei Dezimalstellen angeben, rundet Trident den Wert auf drei Dezimalstellen.

NAS StorageClass mit Autogrow

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-nas-autogrow
  annotations:
    trident.netapp.io/autogrowPolicy: "standard-autogrow"
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-nas"
  fsType: "ext4"
allowVolumeExpansion: true
```

SAN StorageClass mit Autogrow

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: database-storage
  annotations:
    trident.netapp.io/autogrowPolicy: "production-db-policy"
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
  fsType: "ext4"
allowVolumeExpansion: true
```

Autogrow-Richtlinien verwalten

Nachdem Sie Autogrow Policies erstellt haben, können Sie diese nach Bedarf anzeigen, aktualisieren und löschen. Sie können außerdem überwachen, welche Volumes eine bestimmte Policy verwenden.

Autogrow-Richtlinien anzeigen

Alle Richtlinien auflisten

Verwenden Sie `kubectl` , um alle Autogrow Policies in Ihrem Cluster aufzulisten:

```
kubectl get tridentautogrowpolicy
```

Alternativ verwenden Sie `tridentctl`:

```
tridentctl get autogrowpolicy
```

Richtliniendetails anzeigen

Um die vollständige Spezifikation und den Status einer Richtlinie anzuzeigen:

```
kubectl describe tridentautogrowpolicy production-db-policy
```

So zeigen Sie eine Richtlinie mit den zugehörigen Volumes im YAML-Format an:

```
tridentctl get autogrowpolicy production-db-policy -o yaml
```

Eine Autogrow-Richtlinie aktualisieren

Sie können eine bestehende Richtlinie anpassen, um deren Schwellenwert, Wachstumsbetrag oder maximale Größe zu ändern. Änderungen werden sofort für alle Volumes wirksam, die diese Richtlinie verwenden.

WICHTIG	Änderungen betreffen alle Volumes, die derzeit die Richtlinie verwenden. Testen Sie Änderungen nach Möglichkeit zuerst in einer Nicht-Produktionsumgebung.

Schritte

1. Richtlinie bearbeiten:

   ```
   kubectl edit tridentautogrowpolicy production-db-policy
   ```

2. Ändern Sie die `spec` Felder nach Bedarf:

```
spec:
  usedThreshold: "75%"     # Changed from 80%
  growthAmount: "20%"      # Changed from 10%
  maxSize: "1Ti"           # Changed from 500Gi
```

3. Speichern und beenden. Die Änderungen werden sofort wirksam.

Aktualisierungsüberlegungen

- **Sofortige Wirkung:** Alle Volumes, die die Richtlinie nutzen, übernehmen bei der nächsten Wachstumsbewertung die neuen Parameter.
- **Kein Neustart des Volumes erforderlich:** Änderungen gelten für den nächsten Wachstumsvorgang.
- **Erst testen:** Validieren Sie Änderungen nach Möglichkeit in einer Nicht-Produktionsumgebung.
- **Änderungen kommunizieren:** Benachrichtigen Sie die Teams, wenn Sie gemeinsam genutzte Richtlinien ändern.

Löschen einer Autogrow-Richtlinie

Autogrow-Richtlinien verwenden Finalizer-Schutz, um eine versehentliche Löschung zu verhindern, während Volumes sie aktiv verwenden.

Schritte

1. Löschen Sie die Richtlinie:

```
kubectl delete tridentautogrowpolicy production-db-policy
```

2. Wenn Volumes die Richtlinie noch verwenden, wechselt die Löschung in einen `Deleting` Status. Prüfen Sie, welche Volumes betroffen sind:

```
tridentctl get autogrowpolicy production-db-policy -o yaml
```

3. Entfernen Sie die Richtlinie von jedem betroffenen Volume. Wählen Sie eine der folgenden Optionen:

   - **Option A: Automatisches Wachstum explizit deaktivieren**, indem Sie die Annotation auf `"none"` setzen:

     ```
     kubectl annotate pvc <pvc-name> \
       trident.netapp.io/autogrowPolicy="none" \
       --overwrite
     ```

   - **Option B: Die Annotation vollständig entfernen:**

```
kubectl annotate pvc <pvc-name> \
  trident.netapp.io/autogrowPolicy-
```

Löschverhalten

Szenario	Verhalten
Keine Volumes verwenden die Richtlinie	Die Richtlinie wird sofort gelöscht.
Volumes verwenden die Richtlinie	Die Richtlinie tritt in `Deleting` Zustand ein. Ein Finalisierer blockiert den Abschluss, bis alle Volumes entfernt sind.
Alle Volumes werden aus der Richtlinie entfernt	Finalisierer werden entfernt und die Richtlinie wird gelöscht.

Autogrow-Richtliniennutzung überwachen

Überprüfen Sie Volumes mithilfe einer Richtlinie

```
tridentctl get autogrowpolicy production-db-policy -o json | jq '.volumes'
```

Ermitteln Sie, welche Richtlinie ein Volume verwendet

```
kubectl get pvc database-pvc -o
jsonpath='{.metadata.annotations.trident\.netapp\.io/autogrowPolicy}'
```

Richtlinienereignisse überwachen

```
kubectl get events --field-selector
involvedObject.kind=TridentAutogrowPolicy
```

Unterstützte Protokolle

Autogrow unterstützt die folgenden Speicherprotokolle:

- NFS
- iSCSI
- FCP
- NVMe

HINWEIS	Für SAN-Volumes, wenn die konfigurierte `growthAmount` 50 MiB oder weniger beträgt, erhöht Trident den Wachstumsbetrag für die Größenänderung automatisch auf 51 MB, solange die resultierende Größe `maxSize` nicht überschreitet.

Bekannte Einschränkungen

- **ONTAP NVMe raw block volumes:** Volumes, die mit ONTAP Versionen vor 9.16.1 erstellt wurden, unterstützen kein automatisches Wachstum.
- **Vorhandene Volumes (Brownfield-Bereitstellungen):** Die automatische Vergrößerung funktioniert möglicherweise nicht für vorhandene Volumes, selbst wenn eine gültige Autogrow Policy angewendet wird. Dies liegt an einer laufenden Migration der Volume-Veröffentlichungen. Um zu bestätigen, dass die Migration abgeschlossen ist, überprüfen Sie die Trident Controller-Protokolle auf "`Migration completed`" Meldungen.

Häufig gestellte Fragen

Wann wertet Trident den Schwellenwert aus?

Trident überwacht kontinuierlich die Speicherauslastung. Wenn die belegte Kapazität die `usedThreshold`überschreitet, erstellt Trident eine interne Größenänderungsanforderung und erweitert das Volume um die konfigurierte `growthAmount`.

Diese Richtlinie löst beispielsweise eine Erweiterung bei 80 % Kapazität aus und erhöht das Volumen jedes Mal um 10 %, bis zu einem Maximum von 500 GiB:

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: standard-autogrow
spec:
  usedThreshold: "80%"
  growthAmount: "10%"
  maxSize: "500Gi"
```

Kann ich eine Richtlinie anwenden, nachdem Volumes bereits bereitgestellt wurden?

Ja. Sie können jederzeit eine Autogrow-Richtlinie erstellen und diese auf bestehende PVCs anwenden, indem Sie die `trident.netapp.io/autogrowPolicy` Annotation hinzufügen oder aktualisieren. Sie müssen weder die PVC noch die StorageClass neu erstellen.

So wenden Sie eine Richtlinie auf ein bestehendes PVC an:

```
kubectl annotate pvc <pvc-name> \
  trident.netapp.io/autogrowPolicy="production-db-policy" \
  --overwrite
```

Um eine Richtlinie auf eine bestehende StorageClass anzuwenden:

```
kubectl annotate storageclass ontap-gold \
  trident.netapp.io/autogrowPolicy="production-db-policy" \
  --overwrite
```

Was passiert, wenn ich sowohl für die StorageClass als auch für das PVC eine Autogrow-Richtlinie einrichte?

Die PVC-Annotation hat immer Vorrang. Wenn eine PVC die `trident.netapp.io/autogrowPolicy` Annotation enthält, verwendet Trident diesen Wert unabhängig davon, was die StorageClass angibt. Siehe "Richtlinienvorrang" für Details.

Zum Beispiel, gegeben diese StorageClass:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-gold
  annotations:
    trident.netapp.io/autogrowPolicy: "standard-agp"
provisioner: csi.trident.netapp.io
allowVolumeExpansion: true
```

Und dieses PVC, das die StorageClass-Richtlinie außer Kraft setzt:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: database-pvc
  annotations:
    trident.netapp.io/autogrowPolicy: "logs-policy"
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 50Gi
  storageClassName: ontap-gold
```

Trident verwendet `logs-policy` für `database-pvc`, nicht `standard-agp`.

Wie deaktiviere ich Autogrow für ein bestimmtes Volume?

Setzen Sie die PVC-Annotation auf `"none"`. Dadurch wird jede StorageClass-Richtlinie für dieses Volume überschrieben:

```
kubectl annotate pvc <pvc-name> \
  trident.netapp.io/autogrowPolicy="none" \
  --overwrite
```

Sie können überprüfen, ob autogrow deaktiviert ist:

```
kubectl get pvc <pvc-name> -o jsonpath
='{.metadata.annotations.trident\.netapp\.io/autogrowPolicy}'
```

Erwartete Ausgabe

```
none
```

Was passiert, wenn ein Volumen maxSize erreicht?

Trident stellt die Erweiterung des Volumes ein. Es werden keine weiteren Größenänderungsanforderungen für dieses Volume erstellt, selbst wenn die Nutzung weiterhin über den `usedThreshold` hinaus ansteigt.

Beispielsweise stoppt Trident mit dieser Richtlinie das Wachstum des Volumens, sobald es 100 GiB erreicht:

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: capped-policy
spec:
  usedThreshold: "90%"
  growthAmount: "10Gi"
  maxSize: "100Gi"
```

Um unbegrenztes Wachstum zu ermöglichen, lassen Sie `maxSize` weg oder setzen Sie es auf `0`:

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: unlimited-policy
spec:
  usedThreshold: "85%"
  growthAmount: "10%"
```

Kann ich eine Richtlinie ändern, ohne die Volumes neu zu starten?

Ja. Wenn Sie eine Richtlinie aktualisieren, übernehmen alle Volumes, die diese Richtlinie verwenden, die neuen Parameter bei der nächsten Wachstumsbewertung. Ein Neustart der Volumes ist nicht erforderlich.

Um eine Richtlinie zu aktualisieren:

```
kubectl edit tridentautogrowpolicy production-db-policy
```

Passen Sie die Felder nach Bedarf an:

```
spec:
  usedThreshold: "75%"     # Changed from 80%
  growthAmount: "20%"      # Changed from 10%
  maxSize: "1Ti"           # Changed from 500Gi
```

Speichern und beenden. Überprüfen Sie die aktualisierte Richtlinie:

```
kubectl get tridentautogrowpolicy production-db-policy
```

Erwartete Ausgabe

```
NAME                   USED THRESHOLD   GROWTH AMOUNT   STATE
production-db-policy   75%              20%             Success
```

Warum befindet sich meine Richtlinie im Status „Fehlgeschlagen“?

Ein `Failed` Status zeigt an, dass die Richtlinienspezifikation Validierungsfehler enthält. Führen Sie den folgenden Befehl aus, um die Fehlerdetails anzuzeigen:

```
kubectl describe tridentautogrowpolicy <policy-name>
```

Häufige Ursachen sind ein ungültiger `usedThreshold` (muss zwischen 1–99 % liegen), ein `growthAmount`, der `maxSize` überschreitet, oder ein ungültiges Kubernetes-Mengenformat. Korrigieren Sie die Spezifikation und wenden Sie sie erneut an.

```
kubectl apply -f autogrow-policy.yaml
```

Warum kann ich eine Richtlinie nicht löschen?

Richtlinien verwenden einen Finalisierungsschutz. Wenn Volumes die Richtlinie noch verwenden, wechselt der Löschvorgang in einen `Deleting` Zustand und wartet, bis alle Volumes aus der Richtlinie entfernt wurden.

Identifizieren Sie die betroffenen Volumes:

```
tridentctl get autogrowpolicy production-db-policy -o yaml
```

Entfernen Sie anschließend die Annotation von jedem PVC:

```
# Option A: Explicitly disable autogrow
kubectl annotate pvc <pvc-name> \
  trident.netapp.io/autogrowPolicy="none" \
  --overwrite

# Option B: Remove the annotation entirely
kubectl annotate pvc <pvc-name> \
  trident.netapp.io/autogrowPolicy-
```

Nachdem alle Volumes entfernt wurden, wird der Finalizer freigegeben und die Richtlinie gelöscht.

Funktioniert autogrow mit allen ONTAP Backends?

Autogrow unterstützt NFS-, iSCSI-, FCP- und NVMe-Protokolle. NVMe Raw Block Volumes erfordern jedoch ONTAP 9.16.1 oder höher.

Bei bestehenden Volumes in Brownfield-Umgebungen muss die Migration der Volume-Veröffentlichung möglicherweise abgeschlossen sein, bevor das automatische Wachstum wirksam wird. Überprüfen Sie den Migrationsstatus, indem Sie die Trident-Controller-Protokolle prüfen:

```
kubectl logs -l app=trident-controller -n trident | grep "Migration
completed"
```

Die folgenden StorageClass-Beispiele zeigen Autogrow für NAS- und SAN-Backends konfiguriert:

NAS-Backend

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-nas-autogrow
  annotations:
    trident.netapp.io/autogrowPolicy: "standard-autogrow"
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-nas"
  fsType: "ext4"
allowVolumeExpansion: true
```

SAN-Backend

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: database-storage
  annotations:
    trident.netapp.io/autogrowPolicy: "production-db-policy"
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
  fsType: "ext4"
allowVolumeExpansion: true
```

Was ist der minimale Zuwachs für SAN-Volumes?

Für SAN-Volumes beträgt die effektive Mindestwachstumsgröße 51 MB. Wenn Sie eine `growthAmount` von 50 MiB oder weniger konfigurieren, erhöht Trident das Wachstum für die Größenänderung automatisch auf 51 MB.

Diese Richtlinie legt beispielsweise ein `growthAmount` von `"40Mi"` fest, aber Trident wendet ein Wachstum von 51 MB für jedes SAN-Volume an, das sie verwendet:

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: san-minimal-policy
spec:
  usedThreshold: "85%"
  growthAmount: "40Mi"
  maxSize: "100Gi"
```

Um diese automatische Anpassung zu vermeiden, setzen Sie `growthAmount` auf einen Wert größer als 50 MiB:

```
apiVersion: trident.netapp.io/v1
kind: TridentAutogrowPolicy
metadata:
  name: san-policy
spec:
  usedThreshold: "85%"
  growthAmount: "100Mi"
  maxSize: "500Gi"
```

Importmengen

Sie können vorhandene Speichervolumes als Kubernetes PV mit `tridentctl import` importieren oder indem Sie einen Persistent Volume Claim (PVC) mit Trident-Importannotationen erstellen.

Überblick und Überlegungen

Sie könnten ein Volume in Trident importieren, um:

- Eine Anwendung containerisieren und ihren vorhandenen Datensatz wiederverwenden
- Verwenden Sie einen Klon eines Datensatzes für eine kurzlebige Anwendung
- Einen ausgefallenen Kubernetes-Cluster wiederherstellen
- Anwendungsdaten während der Notfallwiederherstellung migrieren

Überlegungen

Bevor Sie ein Volume importieren, beachten Sie die folgenden Hinweise.

- Trident kann nur ONTAP-Volumes vom Typ RW (Read-Write) importieren. Volumes vom Typ DP (data protection) sind SnapMirror Ziel-Volumes. Sie sollten die Spiegelungsbeziehung aufheben, bevor Sie das Volume in Trident importieren.
- Wir empfehlen, Volumes ohne aktive Verbindungen zu importieren. Um ein aktiv genutztes Volume zu importieren, klonen Sie das Volume und führen Sie dann den Import durch.

WARNUNG	Dies ist besonders wichtig für Block-Volumes, da Kubernetes die vorherige Verbindung nicht erkennen und leicht ein aktives Volume an einen Pod anhängen könnte. Dies kann zu Datenbeschädigung führen.

- Obwohl `StorageClass` auf einem PVC angegeben werden muss, verwendet Trident diesen Parameter beim Import nicht. Speicherklassen werden während der Volume-Erstellung verwendet, um anhand der Speichereigenschaften aus den verfügbaren Pools auszuwählen. Da das Volume bereits existiert, ist beim Import keine Poolauswahl erforderlich. Daher schlägt der Import nicht fehl, selbst wenn das Volume auf einem Backend oder in einem Pool existiert, der nicht der im PVC angegebenen Speicherklasse entspricht.
- Die vorhandene Volumengröße wird ermittelt und im PVC festgelegt. Nachdem das Volumen vom Speichertreiber importiert wurde, wird das PV mit einer ClaimRef auf das PVC erstellt.
  - Die Rückgewinnungsrichtlinie ist anfänglich auf `retain` im PV festgelegt. Nachdem Kubernetes PVC und PV erfolgreich gebunden hat, wird die Rückgewinnungsrichtlinie aktualisiert, um der Rückgewinnungsrichtlinie der Storage Class zu entsprechen.
  - Wenn die Rückgewinnungsrichtlinie der Storage Class `delete` ist, wird das Speichervolume gelöscht, wenn das PV gelöscht wird.
- Standardmäßig verwaltet Trident die PVC und benennt das FlexVol Volume und die LUN im Backend um. Sie können das `--no-manage` Flag verwenden, um ein nicht verwaltetes Volume zu importieren, und das `--no-rename` Flag, um den Volume-Namen beizubehalten.
  - `--no-manage`* - Wenn Sie das `--no-manage` Flag verwenden, führt Trident während des gesamten Lebenszyklus der Objekte keine zusätzlichen Operationen an der PVC oder PV durch. Das Speichervolume wird beim Löschen der PV nicht gelöscht und andere Operationen wie das Klonen und die Größenänderung von Volumes werden ebenfalls ignoriert.
  - `--no-rename`* - Wenn Sie das `--no-rename` Flag verwenden, behält Trident beim Importieren von

Volumes den vorhandenen Volume-Namen bei und verwaltet den Lebenszyklus der Volumes. Diese Option wird nur für die `ontap-nas`, `ontap-san` (einschließlich ASA r2-Systeme) und `ontap-san-economy` Treiber unterstützt.

TIPP	Diese Optionen sind nützlich, wenn Sie Kubernetes für containerisierte Workloads verwenden möchten, aber ansonsten den Lebenszyklus des Speichervolumens außerhalb von Kubernetes verwalten möchten.

- Dem PVC und PV wird eine Anmerkung hinzugefügt, die zwei Zwecken dient: Sie zeigt an, dass das Volume importiert wurde und ob der PVC und PV verwaltet werden. Diese Anmerkung sollte nicht verändert oder entfernt werden.

Importieren Sie ein Volume

Sie können ein Volume entweder mit `tridentctl import` oder durch das Erstellen einer PVC mit Trident-Importanmerkungen importieren.

HINWEIS	Wenn Sie PVC-Anmerkungen verwenden, müssen Sie `tridentctl` nicht herunterladen oder verwenden, um das Volume zu importieren.

Verwendung von tridentctl

Schritte

1. Erstellen Sie eine PVC-Datei (zum Beispiel `pvc.yaml`), die zur Erstellung des PVC verwendet wird. Die PVC-Datei sollte `name`, `namespace`, `accessModes` und `storageClassName` enthalten. Optional können Sie `unixPermissions` in Ihrer PVC-Definition angeben.

   Das Folgende ist ein Beispiel für eine Mindestspezifikation:

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

   HINWEIS Geben Sie nur die erforderlichen Parameter an. Zusätzliche Parameter wie PV-Name oder Volume-Größe können dazu führen, dass der Importbefehl fehlschlägt.

2. Verwenden Sie den `tridentctl import` Befehl, um den Namen des Trident Backend, das das Volume enthält, und den Namen, der das Volume auf dem Storage eindeutig identifiziert (zum Beispiel: ONTAP FlexVol, Element Volume), anzugeben. Das `-f` Argument ist erforderlich, um den Pfad zur PVC-Datei anzugeben.

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

Verwendung von PVC-Anmerkungen

Schritte

1. Erstellen Sie eine PVC-YAML-Datei (zum Beispiel `pvc.yaml`) mit den erforderlichen Trident-Importannotationen. Die PVC-Datei sollte Folgendes enthalten:

   - `name` and `namespace` in Metadaten
   - `accessModes`, `resources.requests.storage`, und `storageClassName` in Spezifikation
   - Anmerkungen:
     - `trident.netapp.io/importOriginalName`: Volumenname auf dem Backend
     - `trident.netapp.io/importBackendUUID`: Backend-UUID, wo das Volume existiert
     - `trident.netapp.io/notManaged` (*Optional*): Auf `"true"` für nicht verwaltete Volumes festlegen. Standard ist `"false"`.

Im Folgenden finden Sie eine Beispielspezifikation für den Import eines verwalteten Volumes:

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: <pvc-name>
  namespace: <namespace>
  annotations:
    trident.netapp.io/importOriginalName: "<volume-name>"
    trident.netapp.io/importBackendUUID: "<backend-uuid>"
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: <size>
  storageClassName: <storage-class-name>
```

2. Wenden Sie die PVC YAML-Datei auf Ihren Kubernetes-Cluster an:

```
kubectl apply -f <pvc-file>.yaml
```

Trident wird das Volumen automatisch importieren und es an das PVC binden.

Beispiele

Überprüfen Sie die folgenden Beispiele für den Volumenimport für unterstützte Treiber.

ONTAP NAS und ONTAP NAS FlexGroup

Trident unterstützt den Volumenimport mithilfe der `ontap-nas` und `ontap-nas-flexgroup` Treiber.

HINWEIS

- Trident unterstützt den Volumenimport mit dem `ontap-nas-economy` Treiber nicht.
- Die `ontap-nas` und `ontap-nas-flexgroup`Treiber erlauben keine doppelten Datenträgernamen.

Jedes mit dem `ontap-nas` Treiber erstellte Volume ist ein FlexVol Volume auf dem ONTAP Cluster. Das Importieren von FlexVol Volumes mit dem `ontap-nas` Treiber funktioniert genauso. Ein FlexVol Volume, das bereits auf einem ONTAP Cluster existiert, kann als `ontap-nas` PVC importiert werden. Ebenso können FlexGroup Volumes als `ontap-nas-flexgroup` PVCs importiert werden.

ONTAP NAS-Beispiele mit tridentctl

Die folgenden Beispiele zeigen, wie verwaltete und nicht verwaltete Volumes mit `tridentctl` importiert werden.

Verwaltetes Volume

Das folgende Beispiel importiert ein Volume mit dem Namen `managed_volume` auf einem Backend mit dem Namen `ontap_nas`:

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

Unverwaltetes Volume

Bei Verwendung des `--no-manage` Arguments benennt Trident das Volume nicht um.

Das folgende Beispiel importiert `unmanaged_volume` auf dem `ontap_nas` Backend:

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

ONTAP NAS-Beispiele mit PVC-Annotationen

Die folgenden Beispiele zeigen, wie verwaltete und nicht verwaltete Volumes mithilfe von PVC-Annotationen importiert werden.

Verwaltetes Volume

Das folgende Beispiel importiert ein 1GiB `ontap-nas` Volume namens `ontap_volume1` vom Backend `81abcb27-ea63-49bb-b606-0a5315ac5f21` mit RWO-Zugriffsmodus, der mithilfe von PVC-Annotationen festgelegt ist:

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: <managed-imported-volume>
  namespace: <namespace>
  annotations:
    trident.netapp.io/importOriginalName: "ontap_volume1"
    trident.netapp.io/importBackendUUID: "81abcb27-ea63-49bb-b606-
0a5315ac5f21"
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: <storage-class-name>
```

Unverwaltetes Volume

Das folgende Beispiel importiert ein 1Gi `ontap-nas` Volume mit dem Namen `ontap-volume2` vom Backend `34abcb27-ea63-49bb-b606-0a5315ac5f34` mit festgelegtem RWO-Zugriffsmodus mithilfe von PVC-Annotationen:

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: <umanaged-imported-volume>
  namespace: <namespace>
  annotations:
    trident.netapp.io/importOriginalName: "ontap-volume2"
    trident.netapp.io/importBackendUUID: "34abcb27-ea63-49bb-b606-
0a5315ac5f34"
    trident.netapp.io/notManaged: "true"
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: <storage-class-name>
```

ONTAP SAN

Trident unterstützt den Volume-Import mit den `ontap-san` (iSCSI, NVMe/TCP und FC) und `ontap-san-economy`-Treibern.

Trident kann ONTAP SAN FlexVol Volumes importieren, die eine einzelne LUN enthalten. Dies entspricht dem `ontap-san` Treiber, der für jede PVC ein FlexVol Volume erstellt und eine LUN innerhalb des FlexVol Volumes anlegt. Trident importiert das FlexVol Volume und ordnet es der PVC-Definition zu. Trident kann `ontap-san-economy` Volumes importieren, die mehrere LUNs enthalten.

Die folgenden Beispiele zeigen, wie verwaltete und nicht verwaltete Volumes importiert werden:

Verwaltetes Volume

Für verwaltete Volumes benennt Trident das FlexVol Volume in das `pvc-<uuid>` Format um und die LUN innerhalb des FlexVol Volume in `lun0` um.

Das folgende Beispiel importiert das `ontap-san-managed` FlexVol Volume, das sich auf dem `ontap_san_default` Backend befindet:

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

Unverwaltetes Volume

Das folgende Beispiel importiert `unmanaged_example_volume` auf dem `ontap_san` Backend:

```
tridentctl import volume -n trident san_blog unmanaged_example_volume
-f pvc-import.yaml --no-manage
+-----------------------------------------+---------+--------------
+----------+-----------------------------------------+--------+---------+
|                   NAME                  |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID              | STATE  | MANAGED |
+-----------------------------------------+---------+--------------
+----------+-----------------------------------------+--------+---------+
| pvc-1fc999c9-ce8c-459c-82e4-ed4380a4b228 | 1.0 GiB | san-blog      |
block    | e3275890-7d80-4af6-90cc-c7a0759f555a | online | false   |
+-----------------------------------------+---------+--------------
+----------+-----------------------------------------+--------+---------+
```

Wenn Sie LUNs Igroups zugeordnet haben, die denselben IQN wie ein Kubernetes-Knoten-IQN verwenden (siehe folgendes Beispiel), erhalten Sie die Fehlermeldung: `LUN already mapped to initiator(s) in this group`. Sie müssen den Initiator entfernen oder die Zuordnung der LUN aufheben, um das Volume zu importieren.

```
Vserver   Igroup       Protocol OS Type  Initiators
--------- ------------ -------- -------- ------------------------------------
svm0      k8s-nodename.example.com-fe5d36f2-cded-4f38-9eb0-c7719fc2f9f3
                       iscsi    linux    iqn.1994-05.com.redhat:4c2e1cf35e0
svm0      unmanaged-example-igroup
                       mixed    linux    iqn.1994-05.com.redhat:4c2e1cf35e0
```

ONTAP SAN-economy Beispiele

Die folgenden Beispiele zeigen, wie verwaltete und nicht verwaltete Volumes für das `ontap-san-economy` Backend importiert werden.

Verwaltetes Volume

Wenn Sie ein verwaltetes Volume importieren, übernimmt Trident die Eigentümerschaft des FlexVol und benennt es um. Sie müssen diese Umbenennung berücksichtigen, wenn Sie mehrere LUNs aus demselben FlexVol importieren.

Das folgende Beispiel importiert `lun1` aus der FlexVol `toimport` als verwaltetes Volume mit dem Namen `vol-managed-saneco`:

```
tridentctl import volume vol-managed-saneco toimport/lun1 -f
import1.yaml
```

Nach dem Import `lun1` benennt Trident die FlexVol (zum Beispiel in `trident_lun_pool_xyz`) um. Um zusätzliche LUNs aus derselben FlexVol zu importieren, verwenden Sie den neuen FlexVol-Namen:

```
tridentctl import volume vol-managed-saneco trident_lun_pool_xyz/lun2
-f import2.yaml
```

HINWEIS	Das `ontap-san-economy` Backend importiert jeweils eine LUN. Sie können mehrere Importe mithilfe eines Skripts automatisieren.

Unverwaltetes Volume

Wenn Sie ein nicht verwaltetes Volume importieren, übernimmt Trident nicht die Besitzrechte an dem FlexVol. Allerdings müssen das FlexVol und die LUN den Trident-Namenskonventionen entsprechen.

FlexVol-Benennungsformat

```
trident_lun_pool_STORAGEPREFIX_RANDOMSTRING
```

- `STORAGEPREFIX` ist der Wert von `storagePrefix` in Ihrer Backend-Konfiguration. Der Standardwert ist `trident`.
- `RANDOMSTRING` ist eine beliebige Zeichenkette Ihrer Wahl.

LUN-Benennungsanforderung

Der LUN muss mit einem Namen versehen werden `lun0`.

Beispiel

Wenn Ihre `storagePrefix` `xyz` ist, lautet der vollständige Pfad zur LUN:

```
trident_lun_pool_xyz_randomstring/lun0
```

Element

Trident unterstützt NetApp Element Software und NetApp HCI Volume-Import mit dem `solidfire-san`

Treiber.

HINWEIS	Der Element-Treiber unterstützt doppelte Volume-Namen. Trident gibt jedoch einen Fehler zurück, wenn doppelte Volume-Namen vorhanden sind. Als Workaround klonen Sie das Volume, vergeben einen eindeutigen Volume-Namen und importieren das geklonte Volume.

Das folgende Beispiel importiert ein `element-managed` Volume auf Backend `element_default`.

```
tridentctl import volume element_default element-managed -f pvc-basic-
import.yaml -n trident -d

+----------------------------------------+--------+--------------
+----------+----------------------------------------+--------+---------+
|                   NAME                 |  SIZE  | STORAGE CLASS |
PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+----------------------------------------+--------+--------------
+----------+----------------------------------------+--------+---------+
| pvc-970ce1ca-2096-4ecd-8545-ac7edc24a8fe | 10 GiB | basic-element |
block    | d3ba047a-ea0b-43f9-9c42-e38e58301c49 | online | true    |
+----------------------------------------+--------+--------------
+----------+----------------------------------------+--------+---------+
```

Azure NetApp Files

Trident unterstützt den Volumenimport mithilfe des `azure-netapp-files` Treibers.

HINWEIS	Um ein Azure NetApp Files-Volume zu importieren, identifizieren Sie das Volume anhand seines Volume-Pfads. Der Volume-Pfad ist der Teil des Exportpfads des Volumes nach dem `:/`. Wenn der Mount-Pfad beispielsweise `10.0.0.2:/importvol1` ist, ist der Volume-Pfad `importvol1`.

Das folgende Beispiel importiert ein `azure-netapp-files` Volume auf Backend `azurenetappfiles_40517` mit dem Volume-Pfad `importvol1`.

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

Google Cloud NetApp Volumes

Trident unterstützt den Volumenimport mithilfe des `google-cloud-netapp-volumes` Treibers.

Das folgende Beispiel importiert ein Volume auf Backend `backend-tbc-gcnv1` mit dem Volume `testvoleasiaeast1`.

```
tridentctl import volume backend-tbc-gcnv1 "testvoleasiaeast1" -f < path-
to-pvc> -n trident

+----------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
|                   NAME                 |  SIZE   | STORAGE CLASS
| PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+----------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
| pvc-a69cda19-218c-4ca9-a941-aea05dd13dc0 |  10 GiB | gcnv-nfs-sc-
identity | file     | 8c18cdf1-0770-4bc0-bcc5-c6295fe6d837 | online | true
|
+----------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
```

Das folgende Beispiel importiert ein `google-cloud-netapp-volumes` Volume, wenn sich zwei Volumes in derselben Region befinden:

```
tridentctl import volume backend-tbc-gcnv1
"projects/123456789100/locations/asia-east1-a/volumes/testvoleasiaeast1"
-f <path-to-pvc> -n trident

+------------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
|                   NAME                   |  SIZE   | STORAGE CLASS
| PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+------------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
| pvc-a69cda19-218c-4ca9-a941-aea05dd13dc0 |  10 GiB | gcnv-nfs-sc-
identity | file     | 8c18cdf1-0770-4bc0-bcc5-c6295fe6d837 | online | true
|
+------------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
```

Volumennamen und Beschriftungen anpassen

Mit Trident können Sie den von Ihnen erstellten Volumes aussagekräftige Namen und Bezeichnungen zuweisen. Dies hilft Ihnen, Volumes zu identifizieren und sie einfach ihren jeweiligen Kubernetes-Ressourcen (PVCs) zuzuordnen. Sie können außerdem auf Backend-Ebene Vorlagen für benutzerdefinierte Volume-Namen und benutzerdefinierte Bezeichnungen definieren; alle Volumes, die Sie erstellen, importieren oder klonen, halten sich an diese Vorlagen.

Bevor Sie beginnen

Unterstützung für anpassbare Volume-Namen und Labels:

- Volume-Erstellungs-, Import- und Klonvorgänge.
- Im Fall des `ontap-nas-economy`-Treibers entspricht nur der Name des Qtree-Volumes der Namensvorlage.
- Im Falle des `ontap-san-economy` Treibers entspricht nur der LUN-Name der Namensvorlage.

Einschränkungen

- Benutzerdefinierte Volumennamen sind nur mit ONTAP-Treibern für lokale Installationen kompatibel.
- Benutzerdefinierte Bezeichnungen werden nur für die `ontap-san`, `ontap-nas` und `ontap-nas-flexgroup` Treiber unterstützt.
- Benutzerdefinierte Volumennamen gelten nicht für vorhandene Volumen.

Wichtige Verhaltensweisen von anpassbaren Volumennamen

- Tritt aufgrund ungültiger Syntax in einer Namensvorlage ein Fehler auf, schlägt die Backend-Erstellung fehl. Schlägt jedoch die Vorlagenanwendung fehl, wird das Volume gemäß der bestehenden Namenskonvention benannt.
- Das Speicherpräfix ist nicht anwendbar, wenn ein Volume mithilfe einer Namensvorlage aus der Backend-Konfiguration benannt wird. Jeder gewünschte Präfixwert kann direkt zur Vorlage hinzugefügt werden.

Backend-Konfigurationsbeispiele mit Namensvorlage und Labels

Benutzerdefinierte Namensvorlagen können auf Stamm- und/oder Poolebene definiert werden.

Beispiel auf Wurzelebene

```
{
  "version": 1,
  "storageDriverName": "ontap-nas",
  "backendName": "ontap-nfs-backend",
  "managementLIF": "<ip address>",
  "svm": "svm0",
  "username": "<admin>",
  "password": "<password>",
  "defaults": {
    "nameTemplate":
"{{.volume.Name}}_{{.labels.cluster}}_{{.volume.Namespace}}_{{.volume.Requ
estName}}"
  },
  "labels": {
    "cluster": "ClusterA",
    "PVC": "{{.volume.Namespace}}_{{.volume.RequestName}}"
  }
}
```

Beispiel auf Poolebene

```
{
  "version": 1,
  "storageDriverName": "ontap-nas",
  "backendName": "ontap-nfs-backend",
  "managementLIF": "<ip address>",
  "svm": "svm0",
  "username": "<admin>",
  "password": "<password>",
  "useREST": true,
  "storage": [
    {
      "labels": {
        "labelname": "label1",
        "name": "{{ .volume.Name }}"
      },
      "defaults": {
        "nameTemplate": "pool01_{{ .volume.Name }}_{{ .labels.cluster
}}_{{ .volume.Namespace }}_{{ .volume.RequestName }}"
      }
    },
    {
      "labels": {
        "cluster": "label2",
        "name": "{{ .volume.Name }}"
      },
      "defaults": {
        "nameTemplate": "pool02_{{ .volume.Name }}_{{ .labels.cluster
}}_{{ .volume.Namespace }}_{{ .volume.RequestName }}"
      }
    }
  ]
}
```

Beispiele für Namensvorlagen

Beispiel 1:

```
"nameTemplate": "{{ .config.StoragePrefix }}_{{ .volume.Name }}_{{
.config.BackendName }}"
```

Beispiel 2:

```
"nameTemplate": "pool_{{ .config.StoragePrefix }}_{{ .volume.Name }}_{{
slice .volume.RequestName 1 5 }}""
```

Zu berücksichtigende Punkte

1. Bei Datenträgerimporten werden die Bezeichnungen nur dann aktualisiert, wenn der vorhandene Datenträger Bezeichnungen in einem bestimmten Format aufweist. Zum Beispiel: `{"provisioning":{"Cluster":"ClusterA", "PVC": "pvcname"}}`.
2. Bei verwalteten Volume-Importen folgt der Volume-Name der Namensvorlage, die auf der Root-Ebene in der Backend-Definition definiert ist.
3. Trident unterstützt die Verwendung eines Slice-Operators mit dem Speicherpräfix nicht.
4. Falls die Vorlagen keine eindeutigen Volume-Namen ergeben, fügt Trident einige zufällige Zeichen hinzu, um eindeutige Volume-Namen zu erstellen.
5. Wenn der benutzerdefinierte Name für ein NAS-Economy-Volume länger als 64 Zeichen ist, benennt Trident die Volumes gemäß der bestehenden Namenskonvention. Bei allen anderen ONTAP Treibern schlägt der Volume-Erstellungsprozess fehl, wenn der Volume-Name die Namensbeschränkung überschreitet.

Ein NFS-Volume über Namespaces hinweg freigeben

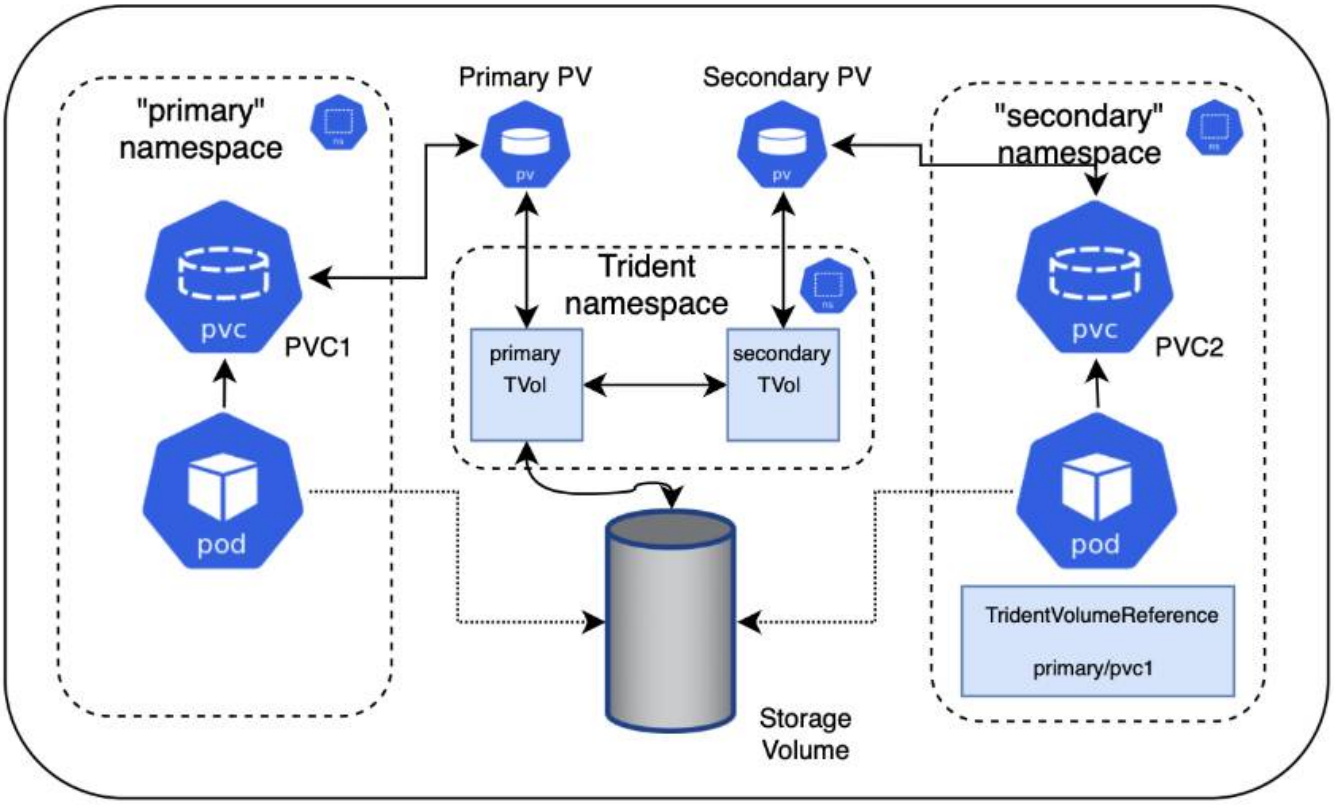
Mit Trident können Sie ein Volume in einem primären Namespace erstellen und es in einem oder mehreren sekundären Namespaces freigeben.

Features

Das TridentVolumeReference CR ermöglicht es Ihnen, ReadWriteMany (RWX)-NFS-Volumes sicher über einen oder mehrere Kubernetes-Namespaces hinweg freizugeben. Diese Kubernetes-native Lösung bietet folgende Vorteile:

- Mehrere Zugriffskontrollebenen zur Gewährleistung der Sicherheit
- Funktioniert mit allen Trident NFS-Volume-Treibern
- Keine Abhängigkeit von tridentctl oder anderen nicht-nativen Kubernetes-Funktionen

Dieses Diagramm veranschaulicht die gemeinsame Nutzung von NFS-Volumes über zwei Kubernetes-Namespaces hinweg.



Schnellstart

Sie können die NFS-Volume-Freigabe in nur wenigen Schritten einrichten.

1 **Konfigurieren Sie den Quell-PVC, um das Volume freizugeben**

Der Inhaber des Quell-Namespace erteilt die Berechtigung zum Zugriff auf die Daten im Quell-PVC.

2 **Erteilen Sie die Berechtigung zum Erstellen eines CR im Ziel-Namespace**

Der Clusteradministrator erteilt dem Eigentümer des Ziel-Namespace die Berechtigung, die TridentVolumeReference CR zu erstellen.

3 **Erstellen Sie TridentVolumeReference im Ziel-Namespace**

Der Eigentümer des Ziel-Namensraums erstellt die TridentVolumeReference CR, um auf die Quell-PVC zu verweisen.

4 **Erstellen Sie den untergeordneten PVC im Ziel-Namespace**

Der Eigentümer des Ziel-Namensraums erstellt den untergeordneten PVC, um die Datenquelle aus dem Quell-PVC zu verwenden.

Konfigurieren Sie die Quell- und Ziel-Namespaces

Um die Sicherheit zu gewährleisten, erfordert die gemeinsame Nutzung über verschiedene Namespaces hinweg die Zusammenarbeit und das Handeln des Quell-Namespace-Inhabers, des Cluster-Administrators und des Ziel-Namespace-Inhabers. Die Benutzerrolle wird in jedem Schritt festgelegt.

Schritte

1. **Quellnamespace-Inhaber:** Erstellen Sie die PVC (`pvc1` im Quellnamespace, die die Berechtigung zur gemeinsamen Nutzung mit dem Zielnamespace (`namespace2` mithilfe der `shareToNamespace` Annotation gewährt.

```
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
```

   Trident erstellt das PV und sein Backend-NFS-Speichervolume.

   HINWEIS

   - Sie können die PVC mithilfe einer durch Kommas getrennten Liste für mehrere Namensräume freigeben. Zum Beispiel `trident.netapp.io/shareToNamespace: namespace2,namespace3,namespace4`.
   - Sie können für alle Namensräume freigeben `*`. Zum Beispiel, `trident.netapp.io/shareToNamespace: *`
   - Sie können die PVC jederzeit aktualisieren, um die `shareToNamespace` Annotation einzuschließen.

2. **Cluster-Administrator:** Stellen Sie sicher, dass die korrekte RBAC-Konfiguration vorhanden ist, um dem Besitzer des Ziel-Namespace die Berechtigung zum Erstellen des TridentVolumeReference CR im Ziel-Namespace zu erteilen.
3. **Ziel-Namespace-Inhaber:** Erstellen Sie eine TridentVolumeReference CR im Ziel-Namespace, die auf den Quell-Namespace verweist `pvc1`.

```
apiVersion: trident.netapp.io/v1
kind: TridentVolumeReference
metadata:
  name: my-first-tvr
  namespace: namespace2
spec:
  pvcName: pvc1
  pvcNamespace: namespace1
```

4. **Ziel-Namespace-Inhaber:** Erstellen Sie einen PVC (`pvc2` im Ziel-Namespace (`namespace2` unter Verwendung der `shareFromPVC`-Annotation, um den Quell-PVC zu kennzeichnen.

```
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
```

HINWEIS Die Größe des Ziel-PVC muss kleiner oder gleich der des Quell-PVC sein.

Ergebnisse

Trident liest die `shareFromPVC` Annotation auf dem Ziel-PVC und erstellt das Ziel-PV als untergeordnetes Volume ohne eigene Speicherressource, das auf das Quell-PV verweist und die Speicherressource des Quell-PV mitnutzt. Das Ziel-PVC und -PV erscheinen wie gewohnt verbunden.

Löschen eines freigegebenen Volumes

Sie können ein Volume löschen, das von mehreren Namespaces gemeinsam genutzt wird. Trident entfernt den Zugriff auf das Volume im Quell-Namespace und erhält den Zugriff für andere Namespaces, die das Volume gemeinsam nutzen. Wenn alle Namespaces, die auf das Volume verweisen, entfernt wurden, löscht Trident das Volume.

Verwenden Sie `tridentctl get` zur Abfrage untergeordneter Volumes

Mit dem [`tridentctl`-Dienstprogramm kann der `get`-Befehl ausgeführt werden, um untergeordnete Volumes abzurufen. Weitere Informationen finden Sie unter `tridentctl` commands and options.

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

Flags:

- `-h, --help`: Hilfe für Bände.
- `--parentOfSubordinate string`: Beschränken Sie die Abfrage auf das untergeordnete Quellvolume.
- `--subordinateOf string`: Beschränken Sie die Abfrage auf untergeordnete Elemente des Volumens.

Einschränkungen

- Trident kann nicht verhindern, dass Ziel-Namespaces auf das gemeinsam genutzte Volume schreiben. Sie sollten Dateisperren oder andere Verfahren verwenden, um das Überschreiben von Daten auf dem gemeinsam genutzten Volume zu verhindern.
- Sie können den Zugriff auf die Quell-PVC nicht widerrufen, indem Sie die `shareToNamespace` oder `shareFromNamespace` Anmerkungen entfernen oder die `TridentVolumeReference` CR löschen. Um den Zugriff zu widerrufen, müssen Sie die untergeordnete PVC löschen.
- Snapshots, Klone und Spiegelungen sind auf untergeordneten Volumes nicht möglich.

Für weitere Informationen

Um mehr über den Volume-Zugriff über Namespaces hinweg zu erfahren:

- Sehen Sie sich die Demo auf "NetAppTV" an.

Volumes über Namespaces hinweg klonen

Mit Trident können Sie neue Volumes erstellen, indem Sie vorhandene Volumes oder Volumesnapshots aus einem anderen Namespace innerhalb desselben Kubernetes-Clusters verwenden.

Voraussetzungen

Bevor Sie Volumes klonen, stellen Sie sicher, dass das Quell- und das Ziel-Backend vom gleichen Typ sind und die gleiche Speicherklasse haben.

HINWEIS	Das Klonen über Namensräume hinweg wird nur für die `ontap-san` und `ontap-nas` Speichertreiber unterstützt. Schreibgeschützte Klone werden nicht unterstützt.

Schnellstart

Das Klonen von Volumes lässt sich in wenigen Schritten einrichten.

1. **Konfigurieren Sie die Quell-PVC, um das Volume zu klonen**

Der Inhaber des Quell-Namespace erteilt die Berechtigung zum Zugriff auf die Daten im Quell-PVC.

2 Erteilen Sie die Berechtigung zum Erstellen eines CR im Ziel-Namespace

Der Clusteradministrator erteilt dem Eigentümer des Ziel-Namespace die Berechtigung, die TridentVolumeReference CR zu erstellen.

3 Erstellen Sie TridentVolumeReference im Ziel-Namespace

Der Eigentümer des Ziel-Namensraums erstellt die TridentVolumeReference CR, um auf die Quell-PVC zu verweisen.

4 Erstellen Sie den Klon-PVC im Ziel-Namespace

Der Eigentümer des Ziel-Namensraums erstellt einen PVC, um den PVC aus dem Quell-Namensraum zu klonen.

Konfigurieren Sie die Quell- und Ziel-Namespaces

Um die Sicherheit zu gewährleisten, erfordert das Klonen von Volumes über Namespaces hinweg die Zusammenarbeit und das Eingreifen des Quell-Namespace-Inhabers, des Cluster-Administrators und des Ziel-Namespace-Inhabers. Die Benutzerrolle wird in jedem Schritt festgelegt.

Schritte

1. **Quellnamespace-Inhaber:** Erstellen Sie die PVC (`pvc1` im Quellnamespace (`namespace1`, die die Berechtigung zur gemeinsamen Nutzung mit dem Zielnamespace (`namespace2` mithilfe der `cloneToNamespace`-Annotation gewährt.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc1
  namespace: namespace1
  annotations:
    trident.netapp.io/cloneToNamespace: namespace2
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: trident-csi
  resources:
    requests:
      storage: 100Gi
```

Trident erstellt das PV und dessen Backend-Speichervolume.

HINWEIS

- Sie können die PVC mithilfe einer durch Kommas getrennten Liste für mehrere Namensräume freigeben. Zum Beispiel `trident.netapp.io/cloneToNamespace: namespace2,namespace3,namespace4`.
- Sie können für alle Namensräume freigeben `*`. Zum Beispiel, `trident.netapp.io/cloneToNamespace: *`
- Sie können die PVC jederzeit aktualisieren, um die `cloneToNamespace` Annotation einzuschließen.

2. **Cluster-Administrator:** Stellen Sie sicher, dass die korrekte RBAC-Konfiguration vorhanden ist, um dem Besitzer des Ziel-Namespace die Berechtigung zum Erstellen des TridentVolumeReference CR im Ziel-Namespace zu erteilen (`namespace2`).
3. **Ziel-Namespace-Inhaber:** Erstellen Sie eine TridentVolumeReference CR im Ziel-Namespace, die auf den Quell-Namespace verweist `pvc1`.

```
apiVersion: trident.netapp.io/v1
kind: TridentVolumeReference
metadata:
  name: my-first-tvr
  namespace: namespace2
spec:
  pvcName: pvc1
  pvcNamespace: namespace1
```

4. **Ziel-Namespace-Inhaber:** Erstellen Sie einen PVC (`pvc2`) im Ziel-Namespace (`namespace2`) mit der Verwendung von `cloneFromPVC` oder `cloneFromSnapshot` und `cloneFromNamespace` Annotationen, um den Quell-PVC zu kennzeichnen.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  annotations:
    trident.netapp.io/cloneFromPVC: pvc1
    trident.netapp.io/cloneFromNamespace: namespace1
  name: pvc2
  namespace: namespace2
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: trident-csi
  resources:
    requests:
      storage: 100Gi
```

Einschränkungen

- Für PVCs, die mit ontap-nas-economy-Treibern bereitgestellt wurden, sind schreibgeschützte Klone nicht unterstützt.

Volumes replizieren mit SnapMirror

Trident unterstützt Spiegelungsbeziehungen zwischen einem Quellvolume in einem Cluster und dem Zielvolume im verbundenen Cluster zur Datenreplikation für die Notfallwiederherstellung. Sie können eine namensbasierte Custom Resource Definition (CRD), genannt Trident Mirror Relationship (TMR), verwenden, um die folgenden Operationen durchzuführen:

- Spiegelbeziehungen zwischen Volumes (PVCs) erstellen
- Spiegelbeziehungen zwischen Volumes entfernen
- Spiegelbeziehungen aufheben
- Das sekundäre Volume während Katastrophenbedingungen (Failover) promoten
- Führen Sie einen verlustfreien Übergang von Anwendungen von Cluster zu Cluster durch (während geplanter Failover oder Migrationen)

Replikationsvoraussetzungen

Stellen Sie sicher, dass die folgenden Voraussetzungen erfüllt sind, bevor Sie beginnen:

ONTAP Cluster

- **Trident**: Trident Version 22.10 oder höher muss auf beiden Quell- und Ziel-Kubernetes-Clustern vorhanden sein, die ONTAP als Backend nutzen.
- **Lizenzen**: ONTAP SnapMirror asynchrone Lizenzen mit dem Data Protection Bundle müssen sowohl auf dem Quell- als auch auf dem Ziel-ONTAP-Cluster aktiviert sein. Weitere Informationen sind unter "SnapMirror Lizenzierungsübersicht in ONTAP" verfügbar.

  Ab ONTAP 9.10.1 werden alle Lizenzen als NetApp Lizenzdatei (NLF) bereitgestellt, bei der es sich um eine einzelne Datei handelt, die mehrere Funktionen aktiviert. Weitere Informationen sind unter "In ONTAP One enthaltene Lizenzen" verfügbar.

HINWEIS	Es wird ausschließlich SnapMirror asynchroner Schutz unterstützt.

Peering

- **Cluster und SVM**: Die ONTAP-Speicher-Backends müssen per Peering verbunden sein. Weitere Informationen sind unter "Cluster- und SVM-Peering-Übersicht" verfügbar.

WICHTIG	Stellen Sie sicher, dass die in der Replikationsbeziehung zwischen zwei ONTAP Clustern verwendeten SVM-Namen eindeutig sind.

- **Trident und SVM**: Die verbundenen Remote-SVMs müssen für Trident auf dem Ziel-Cluster verfügbar sein.

Unterstützte Treiber

NetApp Trident unterstützt die Volume-Replikation mit NetApp SnapMirror-Technologie unter Verwendung von

Storage-Klassen, die von den folgenden Treibern unterstützt werden: **`ontap-nas`: NFS** `ontap-san`: iSCSI **`ontap-san`: FC** `ontap-san`: NVMe/TCP (erfordert mindestens ONTAP Version 9.15.1)

HINWEIS	Die Volume-Replikation mit SnapMirror wird für ASA r2-Systeme nicht unterstützt. Weitere Informationen zu ASA r2-Systemen finden Sie unter "Erfahren Sie mehr über ASA r2-Speichersysteme".

Erstellen Sie ein gespiegeltes PVC

Folgen Sie diesen Schritten und verwenden Sie die CRD-Beispiele, um eine Spiegelbeziehung zwischen primären und sekundären Volumes zu erstellen.

Schritte

1. Führen Sie die folgenden Schritte auf dem primären Kubernetes-Cluster aus:
   a. Erstelle ein StorageClass Objekt mit dem `trident.netapp.io/replication: true` Parameter.

      Beispiel

      ```
      apiVersion: storage.k8s.io/v1
      kind: StorageClass
      metadata:
        name: csi-nas
      provisioner: csi.trident.netapp.io
      parameters:
        backendType: "ontap-nas"
        fsType: "nfs"
        trident.netapp.io/replication: "true"
      ```

   b. Erstellen Sie eine PVC mit zuvor erstellter StorageClass.

      Beispiel

      ```
      kind: PersistentVolumeClaim
      apiVersion: v1
      metadata:
        name: csi-nas
      spec:
        accessModes:
        - ReadWriteMany
        resources:
          requests:
            storage: 1Gi
        storageClassName: csi-nas
      ```

   c. Erstellen Sie eine MirrorRelationship CR mit lokalen Informationen.

Beispiel

```
kind: TridentMirrorRelationship
apiVersion: trident.netapp.io/v1
metadata:
  name: csi-nas
spec:
  state: promoted
  volumeMappings:
  - localPVCName: csi-nas
```

Trident ruft die internen Informationen für das Volume und den aktuellen Datenschutzstatus (DP) des Volumes ab und füllt dann das Statusfeld des MirrorRelationship.

d. Rufen Sie das TridentMirrorRelationship CR ab, um den internen Namen und die SVM des PVC zu erhalten.

```
kubectl get tmr csi-nas
```

```
kind: TridentMirrorRelationship
apiVersion: trident.netapp.io/v1
metadata:
  name: csi-nas
  generation: 1
spec:
  state: promoted
  volumeMappings:
  - localPVCName: csi-nas
status:
  conditions:
  - state: promoted
    localVolumeHandle:
"datavserver:trident_pvc_3bedd23c_46a8_4384_b12b_3c38b313c1e1"
    localPVCName: csi-nas
    observedGeneration: 1
```

2. Führen Sie die folgenden Schritte auf dem sekundären Kubernetes-Cluster aus:
   a. Erstellen Sie eine StorageClass mit dem Parameter trident.netapp.io/replication: true.

Beispiel

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: csi-nas
provisioner: csi.trident.netapp.io
parameters:
  trident.netapp.io/replication: true
```

b. Erstellen Sie ein MirrorRelationship CR mit Ziel- und Quellinformationen.

Beispiel

```
kind: TridentMirrorRelationship
apiVersion: trident.netapp.io/v1
metadata:
  name: csi-nas
spec:
  state: established
  volumeMappings:
  - localPVCName: csi-nas
    remoteVolumeHandle:
"datavserver:trident_pvc_3bedd23c_46a8_4384_b12b_3c38b313c1e1"
```

Trident wird eine SnapMirror Beziehung mit dem konfigurierten Beziehungsrichtliniennamen (oder dem Standardnamen für ONTAP) erstellen und initialisieren.

c. Erstellen Sie eine PVC mit zuvor erstellter StorageClass, um als sekundäres (SnapMirror Ziel) zu fungieren.

Beispiel

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: csi-nas
  annotations:
    trident.netapp.io/mirrorRelationship: csi-nas
spec:
  accessModes:
  - ReadWriteMany
resources:
  requests:
    storage: 1Gi
storageClassName: csi-nas
```

Trident prüft, ob die TridentMirrorRelationship CRD vorhanden ist, und schlägt die Erstellung des Volumes fehl, falls die Beziehung nicht existiert. Wenn die Beziehung besteht, stellt Trident sicher, dass das neue FlexVol volume auf einer SVM platziert wird, die mit der Remote-SVM, die in der MirrorRelationship definiert ist, verbunden ist.

Volumenreplikationszustände

Eine Trident Mirror Relationship (TMR) ist eine CRD, die ein Ende einer Replikationsbeziehung zwischen PVCs darstellt. Die Ziel-TMR hat einen Zustand, der Trident den gewünschten Zustand mitteilt. Die Ziel-TMR hat die folgenden Zustände:

- **Festgestellt**: Das lokale PVC ist das Zielvolume einer SnapMirror Beziehung, und dies ist eine neue Beziehung.
- **Promoted**: Die lokale PVC ist ReadWrite und mountbar, es besteht derzeit keine Spiegelbeziehung.
- **Wiederhergestellt**: Das lokale PVC ist das Zielvolume einer SnapMirror Beziehung und war auch zuvor in dieser SnapMirror Beziehung.
  - Der wiederhergestellte Zustand muss verwendet werden, wenn das Zielvolume jemals in einer Beziehung mit dem Quellvolume stand, da er den Inhalt des Zielvolumes überschreibt.
  - Der wiederhergestellte Zustand schlägt fehl, wenn das Volume zuvor nicht in einer Beziehung mit der Quelle stand.

Sekundäres PVC während eines ungeplanten Failovers promoten

Führen Sie den folgenden Schritt auf dem sekundären Kubernetes-Cluster durch:

- Aktualisiere das Feld *spec.state* von TridentMirrorRelationship auf `promoted`.

Sekundäre PVC während eines geplanten Failovers promoten

Führen Sie während eines geplanten Failovers (Migration) die folgenden Schritte aus, um den sekundären PVC zu promoten:

Schritte

1. Erstellen Sie im primären Kubernetes-Cluster einen Snapshot des PVC und warten Sie, bis der Snapshot erstellt wurde.
2. Erstellen Sie auf dem primären Kubernetes-Cluster die SnapshotInfo CR, um interne Details zu erhalten.

   Beispiel

   ```
   kind: SnapshotInfo
   apiVersion: trident.netapp.io/v1
   metadata:
     name: csi-nas
   spec:
     snapshot-name: csi-nas-snapshot
   ```

3. Aktualisieren Sie auf dem sekundären Kubernetes-Cluster das Feld *spec.state* der *TridentMirrorRelationship* CR auf *promoted* und *spec.promotedSnapshotHandle* auf den internalName des Snapshots.

4. Bestätigen Sie auf dem sekundären Kubernetes-Cluster den Status (status.state-Feld) von TridentMirrorRelationship auf promoted.

Eine Spiegelbeziehung nach einem Failover wiederherstellen

Bevor Sie die Spiegelbeziehung wiederherstellen, wählen Sie die Seite aus, die Sie als neue primäre Seite festlegen möchten.

Schritte

1. Stellen Sie auf dem sekundären Kubernetes-Cluster sicher, dass die Werte für das Feld *spec.remoteVolumeHandle* auf der TridentMirrorRelationship aktualisiert werden.
2. Aktualisieren Sie auf dem sekundären Kubernetes-Cluster das Feld *spec.mirror* von TridentMirrorRelationship auf `reestablished`.

Zusätzliche Operationen

Trident unterstützt die folgenden Operationen auf den primären und sekundären Volumes:

Primäres PVC auf ein neues sekundäres PVC replizieren

Stellen Sie sicher, dass Sie bereits ein primäres PVC und ein sekundäres PVC haben.

Schritte

1. Löschen Sie die PersistentVolumeClaim und TridentMirrorRelationship CRDs aus dem eingerichteten sekundären (Ziel-)Cluster.
2. Löschen Sie die TridentMirrorRelationship CRD aus dem primären (Quell-)Cluster.
3. Erstellen Sie eine neue TridentMirrorRelationship CRD auf dem primären (Quell-)Cluster für das neue sekundäre (Ziel-)PVC, das Sie einrichten möchten.

Ändern Sie die Größe eines gespiegelten, primären oder sekundären PVC

Die PVC-Größe kann wie gewohnt angepasst werden, ONTAP erweitert automatisch alle Ziel-Flevxols, wenn die Datenmenge die aktuelle Größe überschreitet.

Replikation von einem PVC entfernen

Um die Replikation zu entfernen, führen Sie eine der folgenden Operationen auf dem aktuellen sekundären Volume durch:

- Löschen Sie die MirrorRelationship auf dem sekundären PVC. Dadurch wird die Replikationsbeziehung unterbrochen.
- Oder aktualisieren Sie das Feld spec.state auf *promoted*.

Löschen Sie ein PVC (das zuvor gespiegelt wurde)

Trident prüft, ob PVCs repliziert sind, und gibt die Replikationsbeziehung frei, bevor versucht wird, das Volume zu löschen.

Löschen eines TMR

Das Löschen eines TMR auf einer Seite einer Spiegelungsbeziehung führt dazu, dass der verbleibende TMR in den Status *promoted* wechselt, bevor Trident den Löschvorgang abschließt. Befindet sich der zum Löschen ausgewählte TMR bereits im Status *promoted*, besteht keine Spiegelungsbeziehung mehr, der TMR wird

entfernt und Trident stuft den lokalen PVC auf *ReadWrite* hoch. Durch diese Löschung werden SnapMirror-Metadaten für das lokale Volume in ONTAP freigegeben. Wird dieses Volume zukünftig in einer Spiegelungsbeziehung verwendet, muss beim Erstellen der neuen Spiegelungsbeziehung ein neuer TMR mit einem *established* Volume-Replikationsstatus verwendet werden.

Spiegelbeziehungen aktualisieren, wenn ONTAP online ist

Spiegelungsbeziehungen können jederzeit nach ihrer Einrichtung aktualisiert werden. Sie können die Felder `state: promoted` oder `state: reestablished` verwenden, um die Beziehungen zu aktualisieren. Wenn Sie ein Zielvolume zu einem regulären ReadWrite-Volume hochstufen, können Sie mit *promotedSnapshotHandle* einen bestimmten Snapshot angeben, auf den das aktuelle Volume wiederhergestellt werden soll.

Spiegelbeziehungen aktualisieren, wenn ONTAP offline ist

Sie können eine CRD verwenden, um ein SnapMirror-Update durchzuführen, ohne dass Trident eine direkte Verbindung zum ONTAP-Cluster benötigt. Beachten Sie das folgende Beispielformat von TridentActionMirrorUpdate:

Beispiel

```
apiVersion: trident.netapp.io/v1
kind: TridentActionMirrorUpdate
metadata:
  name: update-mirror-b
spec:
  snapshotHandle: "pvc-1234/snapshot-1234"
  tridentMirrorRelationshipName: mirror-b
```

`status.state` spiegelt den Status der TridentActionMirrorUpdate CRD wider. Es kann den Wert *Succeeded*, *In Progress* oder *Failed* annehmen.

CSI-Topologie verwenden

Trident kann selektiv Volumes erstellen und an Knoten in einem Kubernetes-Cluster anhängen, indem es die "CSI-Topologiefunktion" verwendet.

Überblick

Mithilfe der CSI-Topologie-Funktion lässt sich der Zugriff auf Volumes auf eine Teilmenge von Knoten beschränken, basierend auf Regionen und Verfügbarkeitszonen. Cloud-Anbieter ermöglichen es Kubernetes-Administratoren heute, zonenbasierte Knoten zu erstellen. Knoten können sich in verschiedenen Verfügbarkeitszonen innerhalb einer Region oder über verschiedene Regionen hinweg befinden. Um die Bereitstellung von Volumes für Workloads in einer Multi-Zonen-Architektur zu erleichtern, verwendet Trident die CSI-Topologie.

TIPP	Erfahren Sie mehr über die CSI Topology-Funktion "hier,".

Kubernetes bietet zwei einzigartige Volume-Bindungsmodi:

- Mit `VolumeBindingMode` auf `Immediate` gesetzt, erstellt Trident das Volume ohne jegliche

Topologiebewusstheit. Volume-Bindung und dynamische Bereitstellung erfolgen, wenn das PVC erstellt wird. Dies ist die Standardeinstellung `VolumeBindingMode` und eignet sich für Cluster, die keine Topologiebeschränkungen erzwingen. Persistente Volumes werden erstellt, ohne von den Planungsanforderungen des anfordernden Pods abhängig zu sein.

- Mit `VolumeBindingMode` auf `WaitForFirstConsumer` gesetzt, wird die Erstellung und Bindung eines Persistent Volume für eine PVC verzögert, bis ein Pod, der die PVC verwendet, geplant und erstellt wird. So werden Volumes erstellt, um die durch Topologieanforderungen erzwungenen Planungsbeschränkungen zu erfüllen.

HINWEIS	Der `WaitForFirstConsumer` Bindungsmodus benötigt keine Topologiebezeichnungen. Dies kann unabhängig von der CSI-Topologiefunktion verwendet werden.

Was Sie benötigen

Um die CSI Topology nutzen zu können, benötigen Sie Folgendes:

- Ein Kubernetes-Cluster, der einen "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"}
```

- Die Knoten im Cluster sollten Labels aufweisen, die Topologiebewusstsein ermöglichen (`topology.kubernetes.io/region` und `topology.kubernetes.io/zone`). Diese Labels **müssen auf den Knoten im Cluster vorhanden sein**, bevor Trident installiert wird, damit Trident topologiebewusst funktioniert.

```
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 eines topologiebewussten Backends

Trident-Speicher-Backends können so konzipiert werden, dass sie Volumes selektiv basierend auf Verfügbarkeitszonen bereitstellen. Jedes Backend kann einen optionalen `supportedTopologies` Block enthalten, der eine Liste der unterstützten Zonen und Regionen repräsentiert. Für StorageClasses, die ein solches Backend verwenden, würde ein Volume nur erstellt werden, wenn es von einer Anwendung angefordert wird, die in einer unterstützten Region/Zone geplant ist.

Hier ist ein Beispiel für eine Backend-Definition:

YAML

```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

```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"
    }
  ]
}
```

HINWEIS

`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 angegeben werden können. Für StorageClasses, die eine Teilmenge der in einem Backend bereitgestellten Regionen und Zonen enthalten, erstellt Trident ein Volume auf dem Backend.

Sie können `supportedTopologies` pro Speicherpool ebenfalls definieren. Siehe das folgende Beispiel:

```
---
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
    supportedTopologies:
      - topology.kubernetes.io/region: us-central1
        topology.kubernetes.io/zone: us-central1-a
  - labels:
      workload: dev
    supportedTopologies:
      - topology.kubernetes.io/region: us-central1
        topology.kubernetes.io/zone: us-central1-b
```

In diesem Beispiel stehen die `region` und `zone` Bezeichnungen für den Speicherort des Speicherpools. `topology.kubernetes.io/region` und `topology.kubernetes.io/zone` legen fest, von wo aus auf die Speicherpools zugegriffen werden kann.

Schritt 2: Definieren Sie StorageClasses, die topologiebewusst sind

Anhand der den Knoten im Cluster zugewiesenen Topologiebezeichnungen können StorageClasses definiert werden, die Topologieinformationen enthalten. Dadurch werden die Speicherpools bestimmt, die als Kandidaten für PVC-Anfragen dienen, sowie die Teilmenge der Knoten, die die von Trident bereitgestellten Volumes nutzen können.

Siehe das folgende Beispiel:

```
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 wird `volumeBindingMode` auf `WaitForFirstConsumer` gesetzt. PVCs, die mit dieser StorageClass angefordert werden, werden erst verarbeitet, wenn sie in einem Pod referenziert werden. Und `allowedTopologies` gibt die zu verwendenden Zonen und die Region an. Die `netapp-san-us-east1` StorageClass erstellt PVCs auf dem `san-backend-us-east1` oben definierten Backend.

Schritt 3: Eine PVC erstellen und verwenden

Nachdem die StorageClass erstellt und einem Backend zugeordnet wurde, können Sie jetzt PVCs erstellen.

Siehe das Beispiel `spec` unten:

```
---
kind: PersistentVolumeClaim
apiVersion: v1
metadata: null
name: pvc-san
spec: null
accessModes:
  - ReadWriteOnce
resources:
  requests:
    storage: 300Mi
storageClassName: netapp-san-us-east1
```

Das Erstellen eines PVC mit diesem Manifest würde zu Folgendem führen:

```
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
```

Um mit Trident ein Volume zu erstellen und es an das PVC zu binden, verwenden Sie das PVC in einem Pod. Siehe das folgende Beispiel:

```
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
```

Dieses podSpec weist Kubernetes an, den Pod auf Knoten zu planen, die in der `us-east1` Region vorhanden sind, und aus beliebigen Knoten auszuwählen, die in der `us-east1-a` oder `us-east1-b` Zone vorhanden sind.

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
```

Backends aktualisieren, um `supportedTopologies` einzuschließen

Vorhandene Backends können aktualisiert werden, um eine Liste von `supportedTopologies` mit `tridentctl backend update` einzuschließen. Dies hat keine Auswirkungen auf bereits bereitgestellte Volumes und wird nur für nachfolgende PVCs verwendet.

Weitere Informationen

- "Ressourcen für Container verwalten"
- "nodeSelector"
- "Affinität und Anti-Affinität"
- "Taints und Tolerations"

Mit Snapshots arbeiten

Kubernetes-Volume-Snapshots von Persistent Volumes (PVs) ermöglichen zeitpunktgenaue Kopien von Volumes. Sie können einen Snapshot eines mit Trident erstellten Volumes erstellen, einen außerhalb von Trident erstellten Snapshot importieren, ein neues Volume aus einem vorhandenen Snapshot erstellen und Volume-Daten aus Snapshots wiederherstellen.

Überblick

Volume-Snapshot wird von `ontap-nas`, `ontap-nas-flexgroup`, `ontap-san`, `ontap-san-economy`, `solidfire-san`, `azure-netapp-files` und `google-cloud-netapp-volumes` Treibern unterstützt.

Bevor Sie beginnen

Sie benötigen einen externen Snapshot-Controller und benutzerdefinierte Ressourcendefinitionen (CRDs), um mit Snapshots zu arbeiten. Dies ist die Aufgabe des Kubernetes-Orchestrators (zum Beispiel: Kubeadm, GKE, OpenShift).

Falls Ihre Kubernetes-Distribution den Snapshot-Controller und die CRDs nicht enthält, lesen Sie Einen Volume Snapshot Controller bereitstellen.

HINWEIS	Erstellen Sie keinen Snapshot-Controller, wenn Sie in einer GKE-Umgebung bedarfsgesteuerte Volume-Snapshots erstellen. GKE verwendet einen integrierten, versteckten Snapshot-Controller.

Erstellen Sie einen Volume-Snapshot

Schritte

1. Erstellen Sie ein `VolumeSnapshotClass`. Weitere Informationen finden Sie unter "VolumeSnapshotClass".
   - Der `driver` verweist auf den Trident CSI driver.
   - `deletionPolicy` kann `Delete` oder `Retain` sein. Wenn auf `Retain` gesetzt, bleibt der zugrunde liegende physische Snapshot auf dem Speichercluster auch dann erhalten, wenn das `VolumeSnapshot` Objekt gelöscht wird.

     Beispiel

     ```
     cat snap-sc.yaml
     ```

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

2. Erstellen Sie einen Snapshot eines bestehenden PVC.

   Beispiele

   - Dieses Beispiel erstellt eine Snapshot eines bestehenden PVC.

     ```
     cat snap.yaml
     ```

     ```
     apiVersion: snapshot.storage.k8s.io/v1
     kind: VolumeSnapshot
     metadata:
       name: pvc1-snap
     spec:
       volumeSnapshotClassName: csi-snapclass
       source:
         persistentVolumeClaimName: pvc1
     ```

   - Dieses Beispiel erstellt ein Volume-Snapshot-Objekt für eine PVC mit dem Namen `pvc1` und der Name des Snapshots wird auf `pvc1-snap` gesetzt. Ein VolumeSnapshot ist analog zu einer PVC und ist mit einem `VolumeSnapshotContent` Objekt verknüpft, das den eigentlichen Snapshot darstellt.

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

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

- Sie können das `VolumeSnapshotContent` Objekt für das `pvc1-snap` VolumeSnapshot identifizieren, indem Sie es beschreiben. Das `Snapshot Content Name` identifiziert das VolumeSnapshotContent Objekt, das diesem Snapshot dient. Der `Ready To Use` Parameter zeigt an, dass der Snapshot zur Erstellung eines neuen PVC verwendet werden kann.

```
kubectl describe volumesnapshots pvc1-snap
Name:         pvc1-snap
Namespace:    default
...
Spec:
  Snapshot Class Name:    pvc1-snap
  Snapshot Content Name:  snapcontent-e8d8a0ca-9826-11e9-9807-
525400f3f660
  Source:
    API Group:
    Kind:       PersistentVolumeClaim
    Name:       pvc1
Status:
  Creation Time:  2019-06-26T15:27:29Z
  Ready To Use:   true
  Restore Size:   3Gi
...
```

Erstellen Sie eine PVC aus einem Volume-Snapshot

Sie können `dataSource` verwenden, um eine PVC mithilfe eines VolumeSnapshot mit dem Namen `<pvc-name>` als Datenquelle zu erstellen. Nachdem die PVC erstellt wurde, kann sie einem Pod zugeordnet und wie jede andere PVC verwendet werden.

WARNUNG	Die PVC wird im selben Backend wie das Quellvolume erstellt. Siehe "KB: Die Erstellung eines PVC aus einem Trident PVC Snapshot kann in einem alternativen Backend nicht durchgeführt werden".

Im folgenden Beispiel wird die PVC unter Verwendung von `pvc1-snap` als Datenquelle erstellt.

```
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
```

Importieren eines Volume-Snapshots

Trident unterstützt die "Kubernetes pre-provisioned Snapshot-Prozess", damit der Cluster-Administrator ein `VolumeSnapshotContent`-Objekt erstellen und außerhalb von Trident erstellte Snapshots importieren kann.

Bevor Sie beginnen

Trident muss das übergeordnete Volume des Snapshots erstellt oder importiert haben.

Schritte

1. **Cluster-Administrator:** Erstellen Sie ein `VolumeSnapshotContent` Objekt, das auf den Backend-Snapshot verweist. Dadurch wird der Snapshot-Workflow in Trident gestartet.
   - Geben Sie den Namen des Backend-Snapshots in `annotations` als `trident.netapp.io/internalSnapshotName: <"backend-snapshot-name">` an.
   - Geben Sie `<name-of-parent-volume-in-trident>/<volume-snapshot-content-name>` in `snapshotHandle` an. Dies ist die einzige Information, die Trident vom externen Snapshotter im `ListSnapshots`-Aufruf bereitgestellt wird.

HINWEIS	Der `<volumeSnapshotContentName>` kann aufgrund von Namensbeschränkungen der CRs nicht immer mit dem Backend-Snapshot-Namen übereinstimmen.

     Beispiel

     Das folgende Beispiel erstellt ein `VolumeSnapshotContent` Objekt, das auf einen Backend-Snapshot `snap-01` verweist.

```
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshotContent
metadata:
  name: import-snap-content
  annotations:
    trident.netapp.io/internalSnapshotName: "snap-01"  # This is the
name of the snapshot on the backend
spec:
  deletionPolicy: Retain
  driver: csi.trident.netapp.io
  source:
    snapshotHandle: pvc-f71223b5-23b9-4235-bbfe-e269ac7b84b0/import-
snap-content # <import PV name or source PV name>/<volume-snapshot-
content-name>
  volumeSnapshotRef:
    name: import-snap
    namespace: default
```

2. **Cluster-Administrator:** Erstellen Sie das `VolumeSnapshot` CR, das auf das `VolumeSnapshotContent` Objekt verweist. Dadurch wird der Zugriff auf die Verwendung des `VolumeSnapshot` in einem bestimmten Namespace angefordert.

   Beispiel

   Das folgende Beispiel erstellt ein `VolumeSnapshot` CR mit dem Namen `import-snap`, das auf das `VolumeSnapshotContent` mit dem Namen `import-snap-content` verweist.

   ```
   apiVersion: snapshot.storage.k8s.io/v1
   kind: VolumeSnapshot
   metadata:
     name: import-snap
   spec:
     # volumeSnapshotClassName: csi-snapclass (not required for pre-
   provisioned or imported snapshots)
     source:
       volumeSnapshotContentName: import-snap-content
   ```

3. **Interne Verarbeitung (keine Aktion erforderlich):** Der externe Snapshotter erkennt den neu erstellten `VolumeSnapshotContent` und führt den `ListSnapshots`-Aufruf aus. Trident erstellt das `TridentSnapshot`.
   - Der externe Snapshotter setzt das `VolumeSnapshotContent` auf `readyToUse` und das `VolumeSnapshot` auf `true`.
   - Trident kehrt zurück `readyToUse=true`.
4. **Beliebiger Benutzer:** Erstellen Sie ein `PersistentVolumeClaim`, um auf das neue `VolumeSnapshot` zu verweisen, wobei der `spec.dataSource` (oder `spec.dataSourceRef`) Name der `VolumeSnapshot`

Name ist.

Beispiel

Das folgende Beispiel erstellt eine PVC, die auf die `VolumeSnapshot` mit dem Namen `import-snap` verweist.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-from-snap
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: simple-sc
  resources:
    requests:
      storage: 1Gi
  dataSource:
    name: import-snap
    kind: VolumeSnapshot
    apiGroup: snapshot.storage.k8s.io
```

Stellen Sie Volumendaten mithilfe von Snapshots wieder her

Das Snapshot-Verzeichnis ist standardmäßig ausgeblendet, um maximale Kompatibilität von Volumes zu gewährleisten, die mit den `ontap-nas` und `ontap-nas-economy`Treibern bereitgestellt wurden. Aktivieren Sie das `.snapshot`Verzeichnis, um Daten direkt aus Snapshots wiederherzustellen.

Verwenden Sie die ONTAP CLI `volume snapshot restore`, um ein Volume auf einen Zustand wiederherzustellen, der in einem früheren Snapshot aufgezeichnet wurde.

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

HINWEIS	Beim Wiederherstellen einer Snapshot-Kopie wird die bestehende Volume-Konfiguration überschrieben. Änderungen an den Volume-Daten, die nach der Erstellung der Snapshot-Kopie vorgenommen wurden, gehen verloren.

In-Place-Volume-Wiederherstellung aus einem Snapshot

Trident ermöglicht die schnelle, direkte Wiederherstellung von Volumes aus einem Snapshot mithilfe des `TridentActionSnapshotRestore` (TASR) CR. Dieser CR fungiert als imperative Kubernetes-Aktion und bleibt nach Abschluss des Vorgangs nicht erhalten.

Trident unterstützt die Snapshot-Wiederherstellung auf den `ontap-san`, `ontap-san-economy`, `ontap-nas`, `ontap-nas-flexgroup`, `azure-netapp-files`, `google-cloud-netapp-volumes` und `solidfire-san`-Treibern.

Bevor Sie beginnen

Sie müssen ein gebundenes PVC und einen verfügbaren Volume Snapshot haben.

- Überprüfen Sie, ob der PVC-Status gebunden ist.

```
kubectl get pvc
```

- Prüfen Sie, ob der Volume-Snapshot einsatzbereit ist.

```
kubectl get vs
```

Schritte

1. Erstellen Sie den TASR CR. Dieses Beispiel erstellt einen CR für PVC `pvc1` und Volume Snapshot `pvc1-snapshot`.

HINWEIS	Der TASR CR muss sich in einem Namespace befinden, in dem der PVC und der VS existieren.

```
cat tasr-pvc1-snapshot.yaml
```

```
apiVersion: trident.netapp.io/v1
kind: TridentActionSnapshotRestore
metadata:
  name: trident-snap
  namespace: trident
spec:
  pvcName: pvc1
  volumeSnapshotName: pvc1-snapshot
```

2. Wenden Sie die CR an, um aus dem Snapshot wiederherzustellen. Dieses Beispiel stellt aus dem Snapshot `pvc1` wieder her.

```
kubectl create -f tasr-pvc1-snapshot.yaml
```

```
tridentactionsnapshotrestore.trident.netapp.io/trident-snap created
```

Ergebnisse

Trident stellt die Daten aus dem Snapshot wieder her. Sie können den Status der Snapshot-Wiederherstellung überprüfen:

```
kubectl get tasr -o yaml
```

```
apiVersion: trident.netapp.io/v1
items:
- apiVersion: trident.netapp.io/v1
  kind: TridentActionSnapshotRestore
  metadata:
    creationTimestamp: "2023-04-14T00:20:33Z"
    generation: 3
    name: trident-snap
    namespace: trident
    resourceVersion: "3453847"
    uid: <uid>
  spec:
    pvcName: pvc1
    volumeSnapshotName: pvc1-snapshot
  status:
    startTime: "2023-04-14T00:20:34Z"
    completionTime: "2023-04-14T00:20:37Z"
    state: Succeeded
kind: List
metadata:
  resourceVersion: ""
```

HINWEIS

- In den meisten Fällen wird Trident den Vorgang bei einem Fehler nicht automatisch wiederholen. Sie müssen den Vorgang erneut ausführen.
- Kubernetes-Benutzer ohne Administratorrechte müssen möglicherweise vom Administrator die Berechtigung erhalten, einen TASR CR in ihrem Anwendungs-Namespace zu erstellen.

Löschen eines PV mit zugehörigen Snapshots

Beim Löschen eines Persistent Volume mit zugehörigen Snapshots wird das entsprechende Trident volume in den Status „Deleting state“ versetzt. Entfernen Sie die Volume-Snapshots, um das Trident volume zu löschen.

Einen Volume Snapshot Controller bereitstellen

Falls Ihre Kubernetes-Distribution den Snapshot-Controller und die CRDs nicht enthält, können Sie sie wie folgt bereitstellen.

Schritte

1. Erstellen Sie Volume Snapshot-CRDs.

```
cat snapshot-setup.sh
```

```
#!/bin/bash
# Create volume snapshot CRDs
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
6.1/client/config/crd/snapshot.storage.k8s.io_volumesnapshotclasses.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
6.1/client/config/crd/snapshot.storage.k8s.io_volumesnapshotcontents.yam
l
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
6.1/client/config/crd/snapshot.storage.k8s.io_volumesnapshots.yaml
```

2. Erstellen Sie den Snapshot Controller.

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-6.1/deploy/kubernetes/snapshot-
controller/rbac-snapshot-controller.yaml
```

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-6.1/deploy/kubernetes/snapshot-
controller/setup-snapshot-controller.yaml
```

HINWEIS	Öffnen Sie bei Bedarf `deploy/kubernetes/snapshot-controller/rbac-snapshot-controller.yaml` und aktualisieren Sie `namespace` auf Ihren Namespace.

Verwandte Links

- "Volume Snapshots"
- "VolumeSnapshotClass"

Arbeiten mit Volume-Group-Snapshots

Kubernetes-Volume-Gruppen-Snapshots von Persistent Volumes (PVs) NetApp Trident bietet die Möglichkeit, Snapshots mehrerer Volumes (einer Gruppe von Volume-Snapshots) zu erstellen. Dieser Volume-Gruppen-Snapshot stellt Kopien mehrerer Volumes dar, die zum gleichen Zeitpunkt erstellt wurden.

HINWEIS	VolumeGroupSnapshot ist eine Beta-Funktion in Kubernetes mit Beta-APIs. Kubernetes 1.32 ist die Mindestversion, die für VolumeGroupSnapshot erforderlich ist.

Snapshots von Volumengruppen erstellen

Snapshot für Volumengruppen wird mit den folgenden Speichertreibern unterstützt:

- `ontap-san` Treiber – nur für die iSCSI- und FC-Protokolle, nicht für das NVMe/TCP-Protokoll.
- `ontap-san-economy` - nur für das iSCSI-Protokoll.
- `ontap-nas`

HINWEIS	Volume-Gruppen-Snapshot wird für NetApp ASA r2 oder AFX-Speichersysteme nicht unterstützt.

Bevor Sie beginnen

- Stellen Sie sicher, dass Ihre Kubernetes-Version K8s 1.32 oder höher ist.
- Sie benötigen einen externen Snapshot-Controller und benutzerdefinierte Ressourcendefinitionen (CRDs), um mit Snapshots zu arbeiten. Dies ist die Aufgabe des Kubernetes-Orchestrators (zum Beispiel: Kubeadm, GKE, OpenShift).

  Falls Ihre Kubernetes-Distribution den externen Snapshot-Controller und die CRDs nicht enthält, lesen Sie Einen Volume Snapshot Controller bereitstellen.

HINWEIS	Erstellen Sie keinen Snapshot-Controller, wenn Sie in einer GKE-Umgebung bedarfsgesteuerte Snapshots von Volumengruppen erstellen. GKE verwendet einen integrierten, versteckten Snapshot-Controller.

- Setzen Sie im Snapshot-Controller-YAML das `CSIVolumeGroupSnapshot` Feature-Gate auf 'true', um sicherzustellen, dass Volume Group Snapshot aktiviert ist.
- Erstellen Sie die erforderlichen Volume Group Snapshot-Klassen, bevor Sie einen Volume Group Snapshot erstellen.
- Stellen Sie sicher, dass sich alle PVCs/Volumes auf demselben SVM befinden, um VolumeGroupSnapshot erstellen zu können.

Schritte

- Erstellen Sie eine VolumeGroupSnapshotClass, bevor Sie eine VolumeGroupSnapshot erstellen. Weitere Informationen finden sich unter "VolumeGroupSnapshotClass".

```
apiVersion: groupsnapshot.storage.k8s.io/v1beta1
kind: VolumeGroupSnapshotClass
metadata:
  name: csi-group-snap-class
  annotations:
    kubernetes.io/description: "Trident group snapshot class"
driver: csi.trident.netapp.io
deletionPolicy: Delete
```

- Erstellen Sie PVCs mit den erforderlichen Labels unter Verwendung vorhandener Storage Classes, oder fügen Sie diese Labels zu bestehenden PVCs hinzu.

Das folgende Beispiel erstellt die PVC unter Verwendung von `pvc1-group-snap` als Datenquelle und der Bezeichnung `consistentGroupSnapshot: groupA`. Definieren Sie den Bezeichnungsschlüssel und -wert entsprechend Ihren Anforderungen.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc1-group-snap
  labels:
    consistentGroupSnapshot: groupA
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 100Mi
  storageClassName: sc1-1
```

- Erstellen Sie ein VolumeGroupSnapshot mit der gleichen Bezeichnung (`consistentGroupSnapshot: groupA` wie im PVC angegeben.

  Dieses Beispiel erstellt einen Snapshot einer Volumengruppe:

```
apiVersion: groupsnapshot.storage.k8s.io/v1beta1
kind: VolumeGroupSnapshot
metadata:
  name: "vgs1"
  namespace: trident
spec:
  volumeGroupSnapshotClassName: csi-group-snap-class
  source:
    selector:
      matchLabels:
        consistentGroupSnapshot: groupA
```

Volumendaten mithilfe eines Gruppen-Snapshots wiederherstellen

Sie können einzelne Persistent Volumes mithilfe der einzelnen Snapshots wiederherstellen, die als Teil des Volume Group Snapshot erstellt wurden. Sie können den Volume Group Snapshot nicht als Einheit wiederherstellen.

Verwenden Sie die ONTAP CLI `volume snapshot restore`, um ein Volume auf einen Zustand wiederherzustellen, der in einem früheren Snapshot aufgezeichnet wurde.

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

HINWEIS	Beim Wiederherstellen einer Snapshot-Kopie wird die bestehende Volume-Konfiguration überschrieben. Änderungen an den Volume-Daten, die nach der Erstellung der Snapshot-Kopie vorgenommen wurden, gehen verloren.

In-Place-Volume-Wiederherstellung aus einem Snapshot

Trident ermöglicht die schnelle, direkte Wiederherstellung von Volumes aus einem Snapshot mithilfe des `TridentActionSnapshotRestore` (TASR) CR. Dieser CR fungiert als imperative Kubernetes-Aktion und bleibt nach Abschluss des Vorgangs nicht erhalten.

Weitere Informationen sind unter "In-Place-Volume-Wiederherstellung aus einem Snapshot" zu finden.

Löschen eines PV mit zugehörigen Gruppensnapshots

Beim Löschen eines Gruppenvolume-Snapshots:

- Sie können VolumeGroupSnapshots als Ganzes löschen, nicht einzelne Snapshots in der Gruppe.
- Wenn PersistentVolumes gelöscht werden, während ein Snapshot für dieses PersistentVolume existiert, versetzt Trident dieses Volume in den Status „wird gelöscht“, da der Snapshot entfernt werden muss, bevor das Volume sicher entfernt werden kann.
- Wenn ein Klon mithilfe eines gruppierten Snapshots erstellt wurde und die Gruppe anschließend gelöscht werden soll, wird ein Split-on-Clone-Vorgang gestartet und die Gruppe kann erst gelöscht werden, wenn der Split abgeschlossen ist.

Einen Volume Snapshot Controller bereitstellen

Falls Ihre Kubernetes-Distribution den Snapshot-Controller und die CRDs nicht enthält, können Sie sie wie folgt bereitstellen.

Schritte

1. Erstellen Sie 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-
8.2/client/config/crd/groupsnapshot.storage.k8s.io_volumegroupsnapshotcl
asses.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
8.2/client/config/crd/groupsnapshot.storage.k8s.io_volumegroupsnapshotco
ntents.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
8.2/client/config/crd/groupsnapshot.storage.k8s.io_volumegroupsnapshots.
yaml
```

2. Erstellen Sie den Snapshot Controller.

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-8.2/deploy/kubernetes/snapshot-
controller/rbac-snapshot-controller.yaml
```

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-8.2/deploy/kubernetes/snapshot-
controller/setup-snapshot-controller.yaml
```

HINWEIS Öffnen Sie bei Bedarf `deploy/kubernetes/snapshot-controller/rbac-snapshot-controller.yaml` und aktualisieren Sie `namespace` auf Ihren Namespace.

Verwandte Links

- "VolumeGroupSnapshotClass"
- "Volume Snapshots"

Trident verwalten und überwachen

Trident aktualisieren

Trident aktualisieren

Ab der Version 24.02 folgt Trident einem viermonatigen Veröffentlichungszyklus und liefert drei Hauptversionen pro Kalenderjahr. Jede neue Version baut auf den vorherigen Versionen auf und bietet neue Funktionen, Leistungsverbesserungen, Fehlerbehebungen und Verbesserungen. Wir empfehlen Ihnen, mindestens einmal jährlich ein Upgrade durchzuführen, um die neuen Funktionen in Trident zu nutzen.

Überlegungen vor dem Upgrade

Beachten Sie beim Upgrade auf die neueste Version von Trident Folgendes:

- Es sollte nur eine einzige Trident-Instanz über alle Namespaces hinweg in einem Kubernetes-Cluster installiert sein.
- Trident 23.07 und höher erfordert v1-Volume-Snapshots und unterstützt Alpha- oder Beta-Snapshots nicht mehr.
- Beim Upgrade ist es wichtig, dass Sie `parameter.fsType` in `StorageClasses` verwenden, die von Trident genutzt werden. Sie können `StorageClasses` löschen und neu erstellen, ohne bereits vorhandene Volumes zu beeinträchtigen.
  - Dies ist eine **Voraussetzung** für die Durchsetzung "Sicherheitskontexte" für SAN-Volumes.
  - Das sample input Verzeichnis enthält Beispiele, wie `storage-class-basic.yaml.templ` und `storage-class-bronze-default.yaml`.
  - Weitere Informationen finden sich unter "Bekannte Probleme".

Schritt 1: Wählen Sie eine Version

Trident-Versionen folgen einer datumsbasierten `YY.MM` Namenskonvention, wobei „YY“ die letzten beiden Ziffern des Jahres und „MM“ der Monat sind. Dot-Releases folgen einer `YY.MM.X` Konvention, wobei „X“ das Patch-Level ist. Sie wählen die Version aus, auf die Sie aktualisieren möchten, basierend auf der Version, von der Sie aktualisieren.

- Sie können ein direktes Upgrade auf jede Zielversion durchführen, die innerhalb eines Vier-Versions-Fensters Ihrer installierten Version liegt. Beispielsweise können Sie direkt von 24.06 (oder jeder 24.06 dot release) auf 25.06 aktualisieren.
- Wenn Sie von einer Version außerhalb des Vier-Versions-Fensters aktualisieren, führen Sie ein mehrstufiges Upgrade durch. Verwenden Sie die Upgrade-Anweisungen für die "frühere Version" Version, von der Sie aktualisieren, um auf die neueste Version zu aktualisieren, die in das Vier-Versions-Fenster passt. Wenn Sie beispielsweise Version 23.07 verwenden und auf 25.06 aktualisieren möchten:
  a. Erstes Upgrade von 23.07 auf 24.06.
  b. Führen Sie dann ein Upgrade von 24.06 auf 25.06 durch.

HINWEIS	Beim Upgrade mit dem Trident-Operator auf der OpenShift Container Platform sollten Sie auf Trident 21.01.1 oder höher aktualisieren. Der mit 21.01.0 veröffentlichte Trident-Operator enthielt ein bekanntes Problem, das in 21.01.1 behoben wurde. Weitere Einzelheiten finden Sie unter "Details zum Problem auf GitHub".

Schritt 2: Ermitteln Sie die ursprüngliche Installationsmethode

Um festzustellen, welche Version Sie ursprünglich zur Installation von Trident verwendet haben:

1. Verwenden Sie `kubectl get pods -n trident`, um die Pods zu untersuchen.
   - Wenn kein Operator-Pod vorhanden ist, wurde Trident mit `tridentctl` installiert.
   - Falls ein Operator-Pod vorhanden ist, wurde Trident entweder manuell oder mithilfe von Helm mit dem Trident-Operator installiert.
2. Wenn ein Operator-Pod vorhanden ist, verwenden Sie `kubectl describe torc` , um festzustellen, ob Trident mit Helm installiert wurde.
   - Wenn ein Helm-Label vorhanden ist, wurde Trident mit Helm installiert.
   - Falls kein Helm-Label vorhanden ist, wurde Trident manuell mit dem Trident-Operator installiert.

Schritt 3: Wählen Sie eine Upgrade-Methode

Im Allgemeinen sollte für das Upgrade dieselbe Methode verwendet werden wie für die Erstinstallation, jedoch kann "zwischen Installationsmethoden wechseln". Es gibt zwei Möglichkeiten, Trident zu aktualisieren.

- "Upgrade mit dem Trident operator"

TIPP	Wir empfehlen Ihnen, "Den Workflow für das Operator-Upgrade verstehen" vor dem Upgrade mit dem Operator zu überprüfen.

*

Upgrade mit dem Operator

Den Workflow für das Operator-Upgrade verstehen

Bevor Sie den Trident-Operator verwenden, um Trident zu aktualisieren, sollten Sie die Hintergrundprozesse verstehen, die während des Upgrades ablaufen. Dies umfasst Änderungen am Trident-Controller, Controller-Pod und Node-Pods sowie am Node-DaemonSet, die Rolling Updates ermöglichen.

Trident Operator-Upgrade-Handhabung

Einer der vielen "Vorteile der Verwendung des Trident Operators" Möglichkeiten, Trident zu installieren und zu aktualisieren, ist die automatische Verwaltung von Trident- und Kubernetes-Objekten, ohne bestehende eingebundene Volumes zu beeinträchtigen. Auf diese Weise kann Trident Upgrades ohne Ausfallzeiten oder "*laufende Aktualisierungen*" unterstützen. Insbesondere kommuniziert der Trident Operator mit dem Kubernetes-Cluster, um:

- Löschen und erstellen Sie die Trident Controller-Bereitstellung und den Node DaemonSet neu.

- Ersetzen Sie den Trident Controller Pod und die Trident Node Pods durch neue Versionen.
  - Wenn ein Knoten nicht aktualisiert wird, verhindert dies nicht, dass die übrigen Knoten aktualisiert werden.
  - Nur Knoten mit einem laufenden Trident Node Pod können Volumes einbinden.

TIPP	Weitere Informationen zur Trident-Architektur auf dem Kubernetes-Cluster finden Sie unter "Trident-Architektur".

Operator-Upgrade-Workflow

Wenn Sie ein Upgrade mit dem Trident operator starten:

1. Der **Trident Operator**:
   a. Erkennt die aktuell installierte Version von Trident (Version *n*).
   b. Aktualisiert alle Kubernetes-Objekte einschließlich CRDs, RBAC und Trident SVC.
   c. Löscht die Trident Controller-Bereitstellung für Version *n*.
   d. Erstellt die Trident Controller-Bereitstellung für Version *n+1*.
2. **Kubernetes** erstellt Trident Controller Pod für *n+1*.
3. Der **Trident Operator**:
   a. Löscht das Trident Node DaemonSet für *n*. Der Operator wartet nicht auf die Beendigung des Node Pods.
   b. Erstellt das Trident Node Daemonset für *n+1*.
4. **Kubernetes** erstellt Trident Node Pods auf Knoten, auf denen kein Trident Node Pod *n* ausgeführt wird. Dadurch wird sichergestellt, dass sich nie mehr als ein Trident Node Pod, unabhängig von der Version, auf einem Knoten befindet.

Aktualisieren Sie eine Trident-Installation mit Trident operator oder Helm

Sie können Trident mithilfe des Trident-Operators entweder manuell oder mit Helm aktualisieren. Sie können von einer Trident-Operator-Installation auf eine andere Trident-Operator-Installation upgraden oder von einer `tridentctl` Installation auf eine Trident-Operator-Version upgraden. Überprüfen Sie "Wählen Sie eine Upgrade-Methode" vor dem Upgrade einer Trident-Operator-Installation.

Eine manuelle Installation aktualisieren

Sie können von einer Trident-Operatorinstallation mit Clusterumfang auf eine andere Trident-Operatorinstallation mit Clusterumfang aktualisieren. Alle Trident Versionen verwenden einen Operator mit Clusterumfang.

HINWEIS	Um von Trident, das mit dem Namespace-Scoped-Operator installiert wurde (Versionen 20.07 bis 20.10), zu aktualisieren, verwenden Sie die Upgrade-Anweisungen für "Ihre installierte Version" von Trident.

Über diese Aufgabe

Trident stellt eine Bundle-Datei bereit, die Sie verwenden können, um den Operator zu installieren und zugehörige Objekte für Ihre Kubernetes-Version zu erstellen.

- Für Cluster, auf denen Kubernetes 1.25 oder später läuft, verwenden Sie "bundle_post_1_25.yaml".

Bevor Sie beginnen

Stellen Sie sicher, dass Sie einen Kubernetes-Cluster mit "eine unterstützte Kubernetes-Version" verwenden.

Schritte

1. Überprüfen Sie Ihre Trident-Version:

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

2. Aktualisieren Sie die `operator.yaml`, `tridentorchestrator_cr.yaml` und `post_1_25_bundle.yaml` mit den Registry- und Imagepfaden für die Version, auf die Sie aktualisieren (z. B. 25.06), sowie dem korrekten Secret.
3. Löschen Sie den Trident-Operator, der zur Installation der aktuellen Trident Instanz verwendet wurde. Wenn Sie beispielsweise von 25.02 aktualisieren, führen Sie den folgenden Befehl aus:

```
kubectl delete -f 25.02.0/trident-installer/deploy/<bundle.yaml> -n
trident
```

4. Wenn Sie Ihre Erstinstallation mithilfe von `TridentOrchestrator` Attributen angepasst haben, können Sie das `TridentOrchestrator` Objekt bearbeiten, um die Installationsparameter zu ändern. Dies kann beispielsweise Änderungen umfassen, die vorgenommen wurden, um gespiegelte Trident- und CSI-Image-Registries für den Offline-Modus festzulegen, Debug-Protokolle zu aktivieren oder Image-Pull-Secrets anzugeben.
5. Installieren Sie Trident mithilfe der passenden Bundle-YAML-Datei für Ihre Umgebung, wobei *<bundle.yaml>* `bundle_pre_1_25.yaml` oder `bundle_post_1_25.yaml` entsprechend Ihrer Kubernetes-Version ist. Wenn Sie beispielsweise Trident 25.06.0 installieren, führen Sie den folgenden Befehl aus:

```
kubectl create -f 25.06.0/trident-installer/deploy/<bundle.yaml> -n
trident
```

6. Bearbeiten Sie den Trident-Torc, um das Image 25.06.0 einzufügen.

Aktualisieren einer Helm-Installation

Sie können eine Trident Helm-Installation aktualisieren.

WARNUNG	Beim Upgrade eines Kubernetes-Clusters von Version 1.24 auf 1.25 oder höher, auf dem Trident installiert ist, müssen Sie values.yaml aktualisieren, um `excludePodSecurityPolicy` auf `true` zu setzen oder `--set excludePodSecurityPolicy=true` zum `helm upgrade` Befehl hinzuzufügen, bevor Sie das Cluster aktualisieren können.

Wenn Sie Ihren Kubernetes-Cluster bereits von Version 1.24 auf 1.25 aktualisiert haben, ohne den Trident helm zu aktualisieren, schlägt das helm-Upgrade fehl. Damit das helm-Upgrade durchgeführt werden kann, führen Sie diese Schritte als Voraussetzungen aus:

1. Installieren Sie das helm-mapkubeapis-Plugin von https://github.com/helm/helm-mapkubeapis.
2. Führen Sie einen Testlauf für die Trident-Release im Namespace durch, in dem Trident installiert ist. Dies listet die Ressourcen auf, die bereinigt werden.

```
helm mapkubeapis --dry-run trident  --namespace trident
```

3. Führen Sie einen vollständigen Lauf mit helm durch, um die Bereinigung vorzunehmen.

```
helm mapkubeapis trident  --namespace trident
```

Schritte

1. Wenn Sie "Trident mit Helm installiert" haben, können Sie `helm upgrade trident netapp-trident/trident-operator --version 100.2602.0` verwenden, um das Upgrade in einem Schritt durchzuführen. Wenn Sie das Helm-Repository nicht hinzugefügt haben oder es nicht für das Upgrade verwenden können:
   a. Laden Sie die neueste Trident Version von "der Abschnitt *Assets* auf GitHub" herunter.
   b. Verwenden Sie den `helm upgrade` Befehl, wobei `trident-operator-26.02.0.tgz` die Version angibt, auf die Sie aktualisieren möchten.

```
helm upgrade <name> trident-operator-26.02.0.tgz
```

HINWEIS	Wenn Sie bei der Erstinstallation benutzerdefinierte Optionen festlegen (z. B. die Angabe privater, gespiegelter Registries für Trident und CSI-Images), hängen Sie den `helm upgrade` Befehl mit `--set` an, um sicherzustellen, dass diese Optionen in den Upgrade-Befehl aufgenommen werden, andernfalls werden die Werte auf die Standardwerte zurückgesetzt.

2. Führen Sie `helm list` aus, um zu überprüfen, ob sowohl die Chart- als auch die App-Version aktualisiert wurden. Führen Sie `tridentctl logs` aus, um eventuelle Debug-Meldungen einzusehen.

Upgrade von einer `tridentctl` Installation auf Trident operator

Sie können auf die neueste Version des Trident-Operators von einer `tridentctl`Installation aus aktualisieren. Die vorhandenen Backends und PVCs stehen automatisch zur Verfügung.

HINWEIS	Vor dem Wechsel zwischen den Installationsmethoden überprüfen Sie "Wechsel zwischen Installationsmethoden".

Schritte

1. Laden Sie die neueste Trident-Version herunter.

```
# Download the release required [26.02.0]
mkdir 26.02.0
cd 26.02.0
wget
https://github.com/NetApp/trident/releases/download/v26.02.0/trident-
installer-26.02.0.tar.gz
tar -xf trident-installer-26.02.0.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. Den cluster-scoped Operator im selben Namespace bereitstellen.

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

serviceaccount/trident-operator created
clusterrole.rbac.authorization.k8s.io/trident-operator created
clusterrolebinding.rbac.authorization.k8s.io/trident-operator created
deployment.apps/trident-operator created
podsecuritypolicy.policy/tridentoperatorpods created

#Examine the pods in the Trident namespace
NAME                                READY   STATUS    RESTARTS   AGE
trident-controller-79df798bdc-m79dc   6/6     Running   0          150d
trident-node-linux-xrst8              2/2     Running   0          150d
trident-operator-5574dbbc68-nthjv     1/1     Running   0          1m30s
```

4. Erstellen Sie eine `TridentOrchestrator` CR für die Installation von Trident.

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

kubectl create -f deploy/crds/tridentorchestrator_cr.yaml

#Examine the pods in the Trident namespace
NAME                                READY   STATUS    RESTARTS   AGE
trident-csi-79df798bdc-m79dc        6/6     Running   0          1m
trident-csi-xrst8                   2/2     Running   0          1m
trident-operator-5574dbbc68-nthjv   1/1     Running   0          5m41s
```

5. Bestätigen Sie, dass Trident auf die beabsichtigte Version aktualisiert wurde.

```
kubectl describe torc trident | grep Message -A 3

Message:                Trident installed
Namespace:              trident
Status:                 Installed
Version:                v26.02.0
```

Upgrade mit tridentctl

Sie können eine bestehende Trident Installation ganz einfach mit `tridentctl` aktualisieren.

Über diese Aufgabe

Die Deinstallation und Neuinstallation von Trident entspricht einem Upgrade. Wenn Sie Trident deinstallieren, werden die von der Trident-Bereitstellung verwendeten Persistent Volume Claim (PVC) und Persistent Volume (PV) nicht gelöscht. PVs, die bereits bereitgestellt wurden, bleiben verfügbar, während Trident offline ist, und Trident stellt Volumes für alle PVCs bereit, die in der Zwischenzeit erstellt werden, nachdem es wieder online ist.

Bevor Sie beginnen

Überprüfen Sie "Wählen Sie eine Upgrade-Methode" vor dem Upgrade mit `tridentctl`.

Schritte

1. Führen Sie den Deinstallationsbefehl in `tridentctl` aus, um alle Ressourcen, die mit Trident verbunden sind, mit Ausnahme der CRDs und zugehörigen Objekte zu entfernen.

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

2. Installieren Sie Trident erneut. Siehe "Installieren Sie Trident mit tridentctl".

WICHTIG	Unterbrechen Sie den Aktualisierungsvorgang nicht. Stellen Sie sicher, dass das Installationsprogramm vollständig durchläuft.

Trident mit tridentctl verwalten

Das "Trident Installer-Bundle" beinhaltet das `tridentctl` Befehlszeilenprogramm, um einfachen Zugriff auf Trident zu ermöglichen. Kubernetes-Benutzer mit ausreichenden Berechtigungen können es verwenden, um Trident zu installieren oder den Namespace zu verwalten, der den Trident Pod enthält.

Befehle und globale Flags

Sie können `tridentctl help` ausführen, um eine Liste der verfügbaren Befehle für `tridentctl` zu erhalten, oder das `--help`-Flag an einen beliebigen Befehl anhängen, um eine Liste der Optionen und Flags für diesen spezifischen Befehl zu erhalten.

```
tridentctl [command] [--optional-flag]
```

Das Trident `tridentctl`-Dienstprogramm unterstützt die folgenden Befehle und globalen Flags.

Befehle

`create`
Fügen Sie eine Ressource zu Trident hinzu.

`delete`
Entfernen Sie eine oder mehrere Ressourcen aus Trident.

`get`
Holen Sie eine oder mehrere Ressourcen von Trident.

`help`
Hilfe zu jedem Befehl.

`images`
Drucken Sie eine Tabelle der Container-Images, die Trident benötigt.

`import`
Importieren Sie eine vorhandene Ressource in Trident.

`install`
Installieren Sie Trident.

`logs`
Drucken Sie die Protokolle von Trident.

`send`
Senden Sie eine Ressource von Trident.

`uninstall`
Deinstallieren Sie Trident.

`update`
Modifizieren Sie eine Ressource in Trident.

`update backend state`
Backend-Operationen vorübergehend aussetzen.

`upgrade`
Aktualisieren Sie eine Ressource in Trident.

`version`
Geben Sie die Version von Trident aus.

Globale Flags

`-d, --debug`
Debug-Ausgabe.

`-h, --help`
Hilfe für `tridentctl`.

`-k, --kubeconfig string`
Geben Sie den `KUBECONFIG` Pfad an, um Befehle lokal oder von einem Kubernetes-Cluster zu einem anderen auszuführen.

HINWEIS	Alternativ können Sie die `KUBECONFIG` Variable so exportieren, dass sie auf einen bestimmten Kubernetes-Cluster verweist und `tridentctl` Befehle an diesen Cluster ausführen.

`-n, --namespace string`
Namespace der Trident-Bereitstellung.

`-o, --output string`
Ausgabeformat. Eines der folgenden: json|yaml|name|wide|ps (Standard).

`-s, --server string`
Adresse/Port der Trident REST-Schnittstelle.

WARNUNG	Trident REST interface kann so konfiguriert werden, dass sie nur unter 127.0.0.1 (für IPv4) oder [::1] (für IPv6) lauscht und Anfragen beantwortet.

Befehlsoptionen und Flags

erstellen

Verwenden Sie den `create` Befehl, um eine Ressource zu Trident hinzuzufügen.

```
tridentctl create [option]
```

Optionen

`backend`: Fügen Sie ein Backend zu Trident hinzu.

löschen

Verwenden Sie den `delete` Befehl, um eine oder mehrere Ressourcen aus Trident zu entfernen.

```
tridentctl delete [option]
```

Optionen

`backend`: Löschen Sie ein oder mehrere Speicher-Backends aus Trident.
`snapshot`: Löschen Sie einen oder mehrere Volume-Snapshots aus Trident.

`storageclass`: Löschen Sie eine oder mehrere Speicherklassen aus Trident.
`volume`: Löschen Sie ein oder mehrere Speichervolumes aus Trident.

erhalten

Verwenden Sie den `get` Befehl, um eine oder mehrere Ressourcen von Trident abzurufen.

```
tridentctl get [option]
```

Optionen

`backend`: Ein oder mehrere Speicher-Backends von Trident.
`snapshot`: Ein oder mehrere Snapshots von Trident.
`storageclass`: Eine oder mehrere Speicherklassen von Trident.
`volume`: Ein oder mehrere Volumes von Trident.

Flags

`-h`, `--help`: Hilfe für Volumes.
`--parentOfSubordinate string`: Abfrage auf untergeordnetes Quellvolume beschränken.
`--subordinateOf string`: Abfrage auf untergeordnete Volumes beschränken.

Bilder

Verwenden Sie `images`-Flags, um eine Tabelle der von Trident benötigten Container-Images auszugeben.

```
tridentctl images [flags]
```

Flags

`-h`, `--help`: Hilfe für Bilder.
`-v`, `--k8s-version string`: Semantische Version des Kubernetes-Clusters.

Importvolumen

Verwenden Sie den `import volume` Befehl, um ein vorhandenes Volume in Trident zu importieren.

```
tridentctl import volume <backendName> <volumeName> [flags]
```

Aliase

`volume`, `v`

Flags

`-f`, `--filename string`: Pfad zur YAML- oder JSON-PVC-Datei.
`-h`, `--help`: Hilfe für Volume.
`--no-manage`: Nur PV/PVC erstellen. Lebenszyklusverwaltung des Volumes wird nicht vorausgesetzt.

installieren

Verwenden Sie die `install` Flags, um Trident zu installieren.

```
tridentctl install [flags]
```

Flags

`--autosupport-image string`: Das Container-Image für Autosupport-Telemetrie (Standardwert „netapp/trident autosupport:<current-version>“).
`--autosupport-proxy string`: Die Adresse/der Port eines Proxys zum Senden von Autosupport-Telemetrie.
`--enable-node-prep`: Versuch, erforderliche Pakete auf den Nodes zu installieren.
`--generate-custom-yaml`: YAML-Dateien generieren, ohne etwas zu installieren.
`-h, --help`: Hilfe für die Installation.
`--http-request-timeout`: Die HTTP-Anfrage-Zeitüberschreitung für die REST-API des Trident-Controllers überschreiben (Standardwert 1m30s).
`--image-registry string`: Die Adresse/der Port einer internen Image-Registry.
`--k8s-timeout duration`: Die Zeitüberschreitung für alle Kubernetes-Operationen (Standardwert 3m0s).
`--kubelet-dir string`: Der Host-Speicherort des internen Status von kubelet (Standardwert „/var/lib/kubelet“).
`--log-format string`: Das Trident-Protokollierungsformat (text, json) (Standardwert „text“).
`--node-prep`: Ermöglicht Trident, die Nodes des Kubernetes-Clusters für die Verwaltung von Volumes mit dem angegebenen Datenspeicherprotokoll vorzubereiten. **Aktuell ist `iscsi` der einzige unterstützte Wert. Ab OpenShift 4.19 ist die minimale unterstützte Trident-Version für dieses Feature 25.06.1.**
`--pv string`: Der Name des von Trident verwendeten Legacy-PV, stellen Sie sicher, dass dieses nicht existiert (Standardwert „trident“).
`--pvc string`: Der Name des von Trident verwendeten Legacy-PVC, stellen Sie sicher, dass dieses nicht existiert (Standardwert „trident“).
`--silence-autosupport`: Keine Autosupport-Bundles automatisch an NetApp senden (Standardwert true).
`--silent`: Die meiste Ausgabe während der Installation deaktivieren.
`--trident-image string`: Das zu installierende Trident-Image.
`--k8s-api-qps`: Das Queries-per-Second-(QPS)-Limit für Kubernetes-API-Anfragen (Standardwert 100; optional).
`--use-custom-yaml`: Vorhandene YAML-Dateien im Setup-Verzeichnis verwenden.
`--use-ipv6`: IPv6 für die Kommunikation von Trident verwenden.

Protokolle

Verwenden Sie `logs`-Flags, um die Protokolle von Trident auszugeben.

```
tridentctl logs [flags]
```

Flags

`-a, --archive`: Erstellt ein Support-Archiv mit allen Protokollen, sofern nicht anders angegeben.
`-h, --help`: Hilfe für Protokolle.
`-l, --log string`: Anzuzeigendes Trident-Protokoll. Einer von trident|auto|trident-operator|all (Standard "auto").
`--node string`: Der Kubernetes Node-Name, von dem die Node-Pod-Protokolle gesammelt werden sollen.
`-p, --previous`: Ruft die Protokolle der vorherigen Containerinstanz ab, falls vorhanden.
`--sidecars`: Ruft die Protokolle der Sidecar-Container ab.

senden

Verwenden Sie den `send` Befehl, um eine Ressource von Trident zu senden.

```
tridentctl send [option]
```

Optionen

`autosupport`: Senden Sie ein Autosupport-Archiv an NetApp.

deinstallieren

Verwenden Sie `uninstall`-Flags, um Trident zu deinstallieren.

```
tridentctl uninstall [flags]
```

Flags

`-h, --help`: Hilfe zur Deinstallation.
`--silent`: Deaktiviert die meisten Ausgaben während der Deinstallation.

aktualisieren

Verwenden Sie den `update` Befehl, um eine Ressource in Trident zu ändern.

```
tridentctl update [option]
```

Optionen

`backend`: Aktualisieren Sie ein Backend in Trident.

Backend-Status aktualisieren

Verwenden Sie den `update backend state` Befehl, um Backend-Operationen anzuhalten oder fortzusetzen.

```
tridentctl update backend state <backend-name> [flag]
```

Zu berücksichtigende Punkte

- Wenn ein Backend mit einem TridentBackendConfig (tbc) erstellt wird, kann das Backend nicht mit einer `backend.json`-Datei aktualisiert werden.
- Wenn der `userState` in einer tbc festgelegt wurde, kann er nicht mit dem `tridentctl update backend state <backend-name> --user-state suspended/normal`-Befehl geändert werden.
- Um die Möglichkeit wiederzuerlangen, das `userState` über tridentctl zu setzen, nachdem es über tbc gesetzt wurde, muss das `userState` Feld aus dem tbc entfernt werden. Dies kann mit dem `kubectl edit tbc` Befehl durchgeführt werden. Nachdem das `userState` Feld entfernt wurde, können Sie mit dem `tridentctl update backend state` Befehl das `userState` eines Backends ändern.
- Verwenden Sie das `tridentctl update backend state`, um das `userState` zu ändern. Sie können das `userState` auch mit `TridentBackendConfig` oder `backend.json`-Datei aktualisieren; dies löst eine vollständige Neuinitialisierung des Backends aus und kann zeitaufwändig sein.

  Flags

  `-h, --help`: Hilfe zum Backend-Status.
  `--user-state`: Auf `suspended` setzen, um Backend-Operationen anzuhalten. Auf `normal` setzen, um Backend-Operationen fortzusetzen. Wenn auf `suspended` gesetzt:

- `AddVolume` und `Import Volume` sind pausiert.

- `CloneVolume`, `ResizeVolume`, `PublishVolume`, `UnPublishVolume`, `CreateSnapshot`, `GetSnapshot`, `RestoreSnapshot`, `DeleteSnapshot`, `RemoveVolume`, `GetVolumeExternal`, `ReconcileNodeAccess` bleiben verfügbar.

Sie können den Backend-Status auch über das `userState` Feld in der Backend-Konfigurationsdatei `TridentBackendConfig` oder `backend.json` aktualisieren. Weitere Informationen finden Sie unter "Optionen zur Verwaltung von Backends" und "Führen Sie die Backend-Verwaltung mit kubectl durch".

Beispiel:

JSON

Führen Sie diese Schritte aus, um die `userState` mit der `backend.json` Datei zu aktualisieren:

1. Bearbeiten Sie die `backend.json` Datei, um das `userState` Feld mit dem Wert 'suspended' einzufügen.
2. Aktualisieren Sie das Backend mit dem `tridentctl update backend` Befehl und dem Pfad zur aktualisierten `backend.json` Datei.

   Beispiel: `tridentctl update backend -f /<path to backend JSON file>/backend.json -n trident`

```
{
  "version": 1,
  "storageDriverName": "ontap-nas",
  "managementLIF": "<redacted>",
  "svm": "nas-svm",
  "backendName": "customBackend",
  "username": "<redacted>",
  "password": "<redacted>",
  "userState": "suspended"
}
```

YAML

Sie können die tbc nach ihrer Anwendung mit dem `kubectl edit <tbc-name> -n <namespace>` Befehl bearbeiten. Das folgende Beispiel aktualisiert den Backend-Status auf „Angehalten“ mit der `userState: suspended` Option:

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-ontap-nas
spec:
  version: 1
  backendName: customBackend
  storageDriverName: ontap-nas
  managementLIF: <redacted>
  svm: nas-svm
  userState: suspended
  credentials:
    name: backend-tbc-ontap-nas-secret
```

Version

Verwenden Sie `version`Flags, um die Version von `tridentctl` und den laufenden Trident Service auszugeben.

```
tridentctl version [flags]
```

Flags

`--client`: Nur Clientversion (kein Server erforderlich).
`-h, --help`: Hilfe zur Version.

Plugin-Unterstützung

Tridentctl unterstützt Plugins ähnlich wie kubectl. Tridentctl erkennt ein Plugin, wenn der Name der Plugin-Binärdatei dem Schema „tridentctl-<plugin>“ entspricht und sich die Binärdatei in einem Ordner befindet, der in der Umgebungsvariablen PATH aufgeführt ist. Alle erkannten Plugins sind im Plugin-Abschnitt der tridentctl-Hilfe aufgelistet. Optional können Sie die Suche auch einschränken, indem Sie einen Plugin-Ordner in der Umgebungsvariablen TRIDENTCTL_PLUGIN_PATH angeben (Beispiel: `TRIDENTCTL_PLUGIN_PATH=~/tridentctl-plugins/`). Wenn die Variable verwendet wird, durchsucht tridentctl nur den angegebenen Ordner.

Trident überwachen

Trident stellt eine Reihe von Prometheus-Metrikenendpunkten bereit, mit denen Sie die Leistung von Trident überwachen können.

Überblick

Die von Trident bereitgestellten Metriken ermöglichen Ihnen Folgendes:

- Behalten Sie den Zustand und die Konfiguration von Trident im Auge. Sie können überprüfen, wie erfolgreich die Vorgänge ablaufen und ob Trident wie erwartet mit den Backends kommunizieren kann.
- Untersuchen Sie die Backend-Nutzungsinformationen und verstehen Sie, wie viele Volumes auf einem Backend bereitgestellt werden und wie viel Speicherplatz verbraucht wird, und so weiter.
- Pflegen Sie eine Zuordnung der Anzahl der auf den verfügbaren Backends bereitgestellten Volumes.
- Performance verfolgen. Sie können sich ansehen, wie lange Trident benötigt, um mit den Backends zu kommunizieren und Operationen auszuführen.

HINWEIS	Standardmäßig werden die Metriken von Trident am Zielport `8001` am `/metrics` Endpoint bereitgestellt. Diese Metriken sind **standardmäßig aktiviert**, wenn Trident installiert ist. Sie können Trident so konfigurieren, dass die Metriken auch über HTTPS an Port `8444` abgerufen werden.

Was Sie benötigen

- Ein Kubernetes-Cluster mit installiertem Trident.
- Eine Prometheus-Instanz. Dies kann eine "containerisierte Prometheus-Bereitstellung" sein oder Sie können wählen, Prometheus als "native Anwendung" auszuführen.

Schritt 1: Definieren Sie ein Prometheus-Ziel

Sie sollten ein Prometheus-Ziel definieren, um die Metriken zu erfassen und Informationen über die von Trident verwalteten Backends, die von ihm erstellten Volumes und so weiter zu erhalten. Siehe "Prometheus Operator-Dokumentation".

Schritt 2: Erstellen Sie einen Prometheus ServiceMonitor

Um die Trident-Metriken zu nutzen, sollten Sie einen Prometheus ServiceMonitor erstellen, der den `trident-csi` Service überwacht und am `metrics` Port lauscht. Ein Beispiel für einen 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 die von dem `trident-csi` Service zurückgegebenen Metriken ab und sucht gezielt nach dem `metrics` Endpoint des Service. Dadurch ist Prometheus nun so konfiguriert, dass es die Metriken von Trident versteht.

Zusätzlich zu den direkt von Trident verfügbaren Metriken stellt Kubelet viele `kubelet_volume_*` Metriken über seinen eigenen Metrik-Endpunkt bereit. Kubelet kann Informationen über die Volumes, die angehängt sind, sowie über Pods und andere interne Operationen, die es verarbeitet, bereitstellen. Siehe "hier,".

Trident-Metriken über HTTPS abrufen

Um Trident-Metriken über HTTPS (Port 8444) zu empfangen, müssen Sie die ServiceMonitor-Definition um die TLS-Konfiguration ergänzen. Außerdem müssen Sie das `trident-csi` Secret aus dem `trident` Namespace in den Namespace kopieren, in dem Prometheus ausgeführt wird. Dies können Sie mit folgendem Befehl tun:

```
kubectl get secret trident-csi -n trident -o yaml | sed 's/namespace:
trident/namespace: monitoring/' | kubectl apply -f -
```

Ein Beispiel ServiceMonitor für HTTPS-Metriken sieht folgendermaßen 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:
    - interval: 15s
      path: /metrics
      port: https-metrics
      scheme: https
      tlsConfig:
        ca:
          secret:
            key: caCert
            name: trident-csi
        cert:
          secret:
            key: clientCert
            name: trident-csi
        keySecret:
          key: clientKey
          name: trident-csi
        serverName: trident-csi
```

Trident unterstützt HTTPS-Metriken in allen Installationsmethoden: tridentctl, Helm chart und Operator:

- Wenn Sie den `tridentctl install` Befehl verwenden, können Sie das `--https-metrics` Flag übergeben, um HTTPS-Metriken zu aktivieren.
- Wenn Sie das Helm-Chart verwenden, können Sie den `httpsMetrics` Parameter einstellen, um HTTPS-Metriken zu aktivieren.
- Wenn Sie YAML-Dateien verwenden, können Sie das `--https_metrics` Flag zum `trident-main` Container in der `trident-deployment.yaml` Datei hinzufügen.

Schritt 3: Trident-Metriken mit PromQL abfragen

PromQL eignet sich gut zum Erstellen von Ausdrücken, die Zeitreihen- oder Tabellendaten zurückgeben.

Hier sind einige PromQL-Abfragen, die Sie verwenden können:

Holen Sie sich Trident-Gesundheitsinformationen

- **Prozentsatz der HTTP 2XX-Antworten von Trident**

```
(sum (trident_rest_ops_seconds_total_count{status_code=~"2.."} OR on()
vector(0)) / sum (trident_rest_ops_seconds_total_count)) * 100
```

- **Prozentsatz der REST-Antworten von Trident über Statuscode**

```
(sum (trident_rest_ops_seconds_total_count) by (status_code)  / scalar
(sum (trident_rest_ops_seconds_total_count))) * 100
```

- **Durchschnittliche Dauer in ms der von Trident durchgeführten Vorgänge**

```
sum by (operation)
(trident_operation_duration_milliseconds_sum{success="true"}) / sum by
(operation)
(trident_operation_duration_milliseconds_count{success="true"})
```

Informationen zur Trident-Nutzung abrufen

- **Durchschnittliche Volumengröße**

```
trident_volume_allocated_bytes/trident_volume_count
```

- **Von jedem Backend bereitgestellter Gesamtspeicherplatz**

```
sum (trident_volume_allocated_bytes) by (backend_uuid)
```

Individuelle Volumennutzung abrufen

HINWEIS	Dies ist nur aktiviert, wenn auch kubelet-Metriken erfasst werden.

- **Prozentsatz des verwendeten Speicherplatzes für jedes Volume**

```
kubelet_volume_stats_used_bytes / kubelet_volume_stats_capacity_bytes *
100
```

Erfahren Sie mehr über Trident AutoSupport-Telemetrie

Standardmäßig sendet Trident täglich Prometheus-Metriken und grundlegende Backend-Informationen an NetApp.

- Um zu verhindern, dass Trident Prometheus-Metriken und grundlegende Backend-Informationen an NetApp sendet, übergeben Sie das `--silence-autosupport` Flag während der Trident-Installation.
- Trident kann Container-Logs auch auf Anfrage an den NetApp Support senden `tridentctl send autosupport`. Sie müssen Trident auslösen, damit es seine Logs hochlädt. Bevor Sie Logs einreichen, sollten Sie NetApps "Datenschutzrichtlinie" akzeptieren.
- Sofern nicht anders angegeben, ruft Trident die Protokolle der letzten 24 Stunden ab.
- Sie können den Aufbewahrungszeitraum für Protokolle mit dem `--since` Flag festlegen. Zum Beispiel: `tridentctl send autosupport --since=1h`. Diese Informationen werden erfasst und über einen `trident-autosupport` Container gesendet, der zusammen mit Trident installiert wird. Sie können das Container-Image unter "Trident AutoSupport" beziehen.
- Trident AutoSupport sammelt oder überträgt keine personenbezogenen Daten (PII) oder persönlichen Informationen. Es enthält eine "EULA" , die nicht für das Trident-Container-Image selbst gilt. Sie können mehr über das Engagement von NetApp für Datensicherheit und Vertrauen erfahren "hier,".

Ein Beispiel für eine von Trident gesendete Nutzlast sieht folgendermaßen aus:

```
---
items:
  - backendUUID: ff3852e1-18a5-4df4-b2d3-f59f829627ed
    protocol: file
    config:
      version: 1
      storageDriverName: ontap-nas
      debug: false
      debugTraceFlags: null
      disableDelete: false
      serialNumbers:
        - nwkvzfanek_SN
      limitVolumeSize: ""
    state: online
    online: true
```

- Die AutoSupport-Nachrichten werden an NetApps AutoSupport-Endpunkt gesendet. Wenn Sie eine private Registry zum Speichern von Container-Images verwenden, können Sie das `--image-registry`-Flag verwenden.
- Sie können Proxy-URLs auch konfigurieren, indem Sie die Installations-YAML-Dateien generieren. Dies kann erfolgen, indem Sie `tridentctl install --generate-custom-yaml` verwenden, um die

YAML-Dateien zu erstellen, und das `--proxy-url` Argument für den `trident-autosupport` Container in `trident-deployment.yaml` hinzufügen.

Trident-Metriken deaktivieren

Um die Meldung von Metriken **zu deaktivieren**, sollten Sie benutzerdefinierte YAML-Dateien (unter Verwendung des `--generate-custom-yaml` Flags) generieren und diese bearbeiten, um das `--metrics` Flag aus der Ausführung für den `trident-main` Container zu entfernen.

Trident deinstallieren

Sie sollten die gleiche Methode zum Deinstallieren von Trident verwenden, die Sie auch zum Installieren von Trident verwendet haben.

Über diese Aufgabe

- Falls nach einem Upgrade Fehler, Abhängigkeitsprobleme oder ein fehlgeschlagenes bzw. unvollständiges Upgrade auftreten, sollten Sie Trident deinstallieren und die vorherige Version gemäß der entsprechenden Anleitung neu installieren "Version". Dies ist die einzige empfohlene Methode, um auf eine frühere Version *downzugraden*.
- Für ein einfaches Upgrade und eine Neuinstallation entfernt das Deinstallieren von Trident weder die CRDs noch die von Trident erstellten zugehörigen Objekte. Wenn Sie Trident und alle seine Daten vollständig entfernen müssen, lesen Sie "Trident und CRDs vollständig entfernen".

Bevor Sie beginnen

Wenn Sie Kubernetes-Cluster außer Betrieb nehmen, müssen Sie alle Anwendungen löschen, die von Trident erstellte Volumes verwenden, bevor Sie Trident deinstallieren. Dadurch wird sichergestellt, dass PVCs auf den Kubernetes-Knoten entfernt werden, bevor sie gelöscht werden.

Ermitteln Sie die ursprüngliche Installationsmethode

Sie sollten die gleiche Methode zur Deinstallation von Trident verwenden, die Sie auch zur Installation verwendet haben. Überprüfen Sie vor der Deinstallation, welche Version Sie ursprünglich zur Installation von Trident verwendet haben.

1. Verwenden Sie `kubectl get pods -n trident`, um die Pods zu untersuchen.
   - Wenn kein Operator-Pod vorhanden ist, wurde Trident mit `tridentctl` installiert.
   - Falls ein Operator-Pod vorhanden ist, wurde Trident entweder manuell oder mithilfe von Helm mit dem Trident-Operator installiert.
2. Wenn ein Operator-Pod vorhanden ist, verwenden Sie `kubectl describe tproc trident` , um festzustellen, ob Trident mit Helm installiert wurde.
   - Wenn ein Helm-Label vorhanden ist, wurde Trident mit Helm installiert.
   - Falls kein Helm-Label vorhanden ist, wurde Trident manuell mit dem Trident-Operator installiert.

Entfernen einer Trident-Operatorinstallation

Sie können eine Trident-Operator-Installation manuell oder mit Helm deinstallieren.

Manuelle Installation deinstallieren

Wenn Sie Trident mithilfe des Operators installiert haben, können Sie es auf eine der folgenden Arten deinstallieren:

1. **Edit `TridentOrchestrator` CR und Deinstallationsflag setzen**:

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

   Wenn das `uninstall` Flag auf `true` gesetzt ist, deinstalliert der Trident-Operator Trident, entfernt jedoch den TridentOrchestrator selbst nicht. Sie sollten den TridentOrchestrator bereinigen und einen neuen erstellen, wenn Sie Trident erneut installieren möchten.

2. **Löschen `TridentOrchestrator`**: Durch das Entfernen der `TridentOrchestrator` CR, die zur Bereitstellung von Trident verwendet wurde, weisen Sie den Operator an, Trident zu deinstallieren. Der Operator verarbeitet die Entfernung von `TridentOrchestrator` und fährt fort, die Trident-Bereitstellung und das DaemonSet zu entfernen, wobei die im Rahmen der Installation erstellten Trident-Pods gelöscht werden.

   ```
   kubectl delete -f deploy/<bundle.yaml> -n <namespace>
   ```

Helm-Installation deinstallieren

Wenn Sie Trident mit Helm installiert haben, können Sie es mit `helm uninstall` deinstallieren.

```
#List the Helm release corresponding to the 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 einer `tridentctl`Installation

Verwenden Sie den `uninstall`-Befehl in `tridentctl`, um alle Ressourcen, die mit Trident verbunden sind, mit Ausnahme der CRDs und zugehörigen Objekte zu entfernen:

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

Trident für Docker

Voraussetzungen für die Bereitstellung

Sie müssen die erforderlichen Protokollvoraussetzungen auf Ihrem Host installieren und konfigurieren, bevor Sie Trident bereitstellen können.

Überprüfen Sie die Anforderungen

- Vergewissern Sie sich, dass Ihre Bereitstellung alle "Anforderungen" erfüllt.
- Vergewissern Sie sich, dass Sie eine unterstützte Version von Docker installiert haben. Falls Ihre Docker-Version veraltet ist, "installieren oder aktualisieren".

```
docker --version
```

- Vergewissern Sie sich, dass die Protokollvoraussetzungen auf Ihrem Host installiert und konfiguriert sind.

NFS-Tools

Installieren Sie die NFS-Tools mithilfe der Befehle für Ihr Betriebssystem.

RHEL 8+

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

Ubuntu

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

WARNUNG	Starten Sie Ihre Worker-Knoten nach der Installation der NFS-Tools neu, um Fehler beim Anhängen von Volumes an Container zu vermeiden.

iSCSI-Tools

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

RHEL 8+

1. Installieren Sie die folgenden Systempakete:

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

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

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

3. Scanvorgang 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
```

HINWEIS	Stellen Sie sicher, dass `etc/multipath.conf` `find_multipaths no` unter `defaults` enthalten ist.

5. Stellen Sie sicher, dass `iscsid` und `multipathd` ausgeführt werden:

```
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. Prüfen Sie, ob die open-iscsi-Version 2.0.874-5ubuntu2.10 oder höher (für bionic) oder 2.0.874-7.1ubuntu6.1 oder höher (für focal) ist:

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

3. Scanvorgang 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
```

HINWEIS Stellen Sie sicher, dass `etc/multipath.conf` `find_multipaths no` unter `defaults` enthalten ist.

5. Stellen Sie sicher, dass `open-iscsi` und `multipath-tools` aktiviert sind und ausgeführt werden:

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

NVMe-Tools

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

HINWEIS

- NVMe erfordert RHEL 9 oder neuer.
- Falls die Kernelversion Ihres Kubernetes-Knotens zu alt ist oder das NVMe-Paket für Ihre Kernelversion nicht verfügbar ist, müssen Sie möglicherweise die Kernelversion Ihres Knotens auf eine Version mit dem NVMe-Paket aktualisieren.

RHEL 9

```
sudo yum install nvme-cli
sudo yum install linux-modules-extra-$(uname -r)
sudo modprobe nvme-tcp
```

Ubuntu

```
sudo apt install nvme-cli
sudo apt -y install linux-modules-extra-$(uname -r)
sudo modprobe nvme-tcp
```

FC-Tools

Installieren Sie die FC-Tools mithilfe der Befehle für Ihr Betriebssystem.

- Bei Verwendung von Worker-Knoten, auf denen RHEL/Red Hat Enterprise Linux CoreOS (RHCOS) mit FC-PVs ausgeführt werden, geben Sie die `discard` mountOption in der StorageClass an, um die Inline-Speicherplatzfreigabe durchzuführen. Siehe "Red Hat Dokumentation".

RHEL 8+

1. Installieren Sie die folgenden Systempakete:

```
sudo yum install -y lsscsi device-mapper-multipath
```

2. Multipathing aktivieren:

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

HINWEIS Stellen Sie sicher, dass `etc/multipath.conf` `find_multipaths no` unter `defaults` enthalten ist.

3. Stellen Sie sicher, dass `multipathd` läuft:

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

Ubuntu

1. Installieren Sie die folgenden Systempakete:

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

2. 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
```

HINWEIS Stellen Sie sicher, dass `etc/multipath.conf` `find_multipaths no` unter `defaults` enthalten ist.

3. Stellen Sie sicher, dass `multipath-tools` aktiviert ist und ausgeführt wird:

```
sudo systemctl status multipath-tools
```

Trident bereitstellen

Trident für Docker bietet eine direkte Integration in das Docker-Ökosystem für NetApp Speicherplattformen. Es unterstützt die Bereitstellung und Verwaltung von Speicherressourcen von der Speicherplattform zu Docker-Hosts, mit einem Framework zur zukünftigen Hinzufügung weiterer Plattformen.

Mehrere Instanzen von Trident können gleichzeitig auf demselben Host ausgeführt werden. Dies ermöglicht simultane Verbindungen zu mehreren Speichersystemen und Speichertypen sowie die Möglichkeit, den für die Docker-Volumes verwendeten Speicher anzupassen.

Was Sie benötigen

Siehe "Voraussetzungen für die Bereitstellung". Nachdem Sie sichergestellt haben, dass die Voraussetzungen erfüllt sind, sind Sie bereit, Trident bereitzustellen.

Docker-verwaltete Plugin-Methode (Version 1.13/17.03 und später)

HINWEIS	**Bevor Sie beginnen** Wenn Sie Trident vor Docker 1.13/17.03 in der traditionellen Daemon-Methode verwendet haben, stellen Sie sicher, dass Sie den Trident-Prozess stoppen und Ihren Docker daemon neu starten, bevor Sie die verwaltete Plugin-Methode verwenden.

1. Alle laufenden Instanzen stoppen:

```
pkill /usr/local/bin/netappdvp
pkill /usr/local/bin/trident
```

2. Starten Sie Docker neu.

```
systemctl restart docker
```

3. Stellen Sie sicher, dass Sie Docker Engine 17.03 (new 1.13) oder höher installiert haben.

```
docker --version
```

Falls 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`, jedoch können Sie jeden beliebigen Namen verwenden, indem Sie die `config` Option zusammen mit dem Dateinamen angeben. Die Konfigurationsdatei muss sich im `/etc/netappdvp` Verzeichnis auf dem Hostsystem befinden.
   - `log-level`: Geben Sie die Protokollierungsebene (`debug`, `info`, `warn`, `error`, `fatal`) an. Der Standardwert ist `info`.

- `debug`: Gibt an, ob die Debug-Protokollierung aktiviert ist. Standardmäßig ist false. Überschreibt die Protokollierungsstufe, wenn true.

  i. Erstellen Sie einen Speicherort für die Konfigurationsdatei:

  ```
  sudo mkdir -p /etc/netappdvp
  ```

  ii. Erstellen Sie die Konfigurationsdatei:

  ```
  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 Trident über das verwaltete Plugin-System. Ersetzen Sie `<version>` durch die von Ihnen verwendete Plugin-Version (xxx.xx.x).

```
docker plugin install --grant-all-permissions --alias netapp
netapp/trident-plugin:<version> config=myConfigFile.json
```

3. Beginnen Sie mit der Nutzung von Trident, um Speicher vom konfigurierten System zu verwenden.

   a. Erstellen Sie ein Volume mit dem Namen "firstVolume":

   ```
   docker volume create -d netapp --name firstVolume
   ```

   b. Erstellen Sie ein Standardvolume, wenn der Container startet:

   ```
   docker run --rm -it --volume-driver netapp --volume
   secondVolume:/my_vol alpine ash
   ```

   c. Entfernen Sie das Volume "firstVolume":

```
docker volume rm firstVolume
```

Traditionelle Methode (Version 1.12 oder früher)

Bevor Sie beginnen

1. Stellen Sie sicher, dass Sie Docker Version 1.10 oder später verwenden.

   ```
   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. Installieren und konfigurieren Sie das NetApp Docker Volume Plugin:
   a. Laden Sie die Anwendung herunter und entpacken Sie sie:

      ```
      wget
      https://github.com/NetApp/trident/releases/download/10.0/trident-
      installer-26.02.0.tar.gz
      tar zxf trident-installer-26.02.0.tar.gz
      ```

   b. Wechseln Sie zu einem Ort im Binärpfad:

      ```
      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. Erstellen Sie einen Speicherort für die Konfigurationsdatei:

      ```
      sudo mkdir -p /etc/netappdvp
      ```

   d. Erstellen Sie die Konfigurationsdatei:

      ```
      cat << EOF > /etc/netappdvp/ontap-nas.json
      ```

```
{
  "version": 1,
  "storageDriverName": "ontap-nas",
  "managementLIF": "10.0.0.1",
  "dataLIF": "10.0.0.2",
  "svm": "svm_nfs",
  "username": "vsadmin",
  "password": "password",
  "aggregate": "aggr1"
}
EOF
```

2. Nachdem Sie die Binärdatei platziert und die Konfigurationsdatei erstellt haben, starten Sie den Trident Daemon mit der gewünschten Konfigurationsdatei.

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

HINWEIS Sofern nicht anders angegeben, lautet der Standardname für den Volume-Treiber "netapp".

Nach dem Start des Daemons können Sie Volumes mit der Docker CLI Schnittstelle erstellen und verwalten.

3. Erstellen Sie ein Volume:

```
docker volume create -d netapp --name trident_1
```

4. Stellen Sie ein Docker-Volume beim Starten eines Containers bereit:

```
docker run --rm -it --volume-driver netapp --volume trident_2:/my_vol
alpine ash
```

5. Entfernen Sie ein Docker-Volume:

```
docker volume rm trident_1
```

```
docker volume rm trident_2
```

Trident beim Systemstart starten

Eine Beispiel-Unit-Datei für systemd-basierte Systeme finden Sie `contrib/trident.service.example` im Git-Repository. Um die Datei unter RHEL zu verwenden, gehen Sie wie folgt vor:

1. Kopieren Sie die Datei an den richtigen Speicherort.

   Sie sollten eindeutige Namen für die Unit-Dateien verwenden, wenn Sie mehr als eine Instanz ausführen.

   ```
   cp contrib/trident.service.example
   /usr/lib/systemd/system/trident.service
   ```

2. Bearbeiten Sie die Datei, ändern Sie die Beschreibung (Zeile 2) so, dass sie dem Treibernamen entspricht, und den Pfad der Konfigurationsdatei (Zeile 9) so, dass er Ihre Umgebung widerspiegelt.
3. Laden Sie systemd neu, damit die Änderungen übernommen werden:

   ```
   systemctl daemon-reload
   ```

4. Aktivieren Sie den Dienst.

   Dieser Name variiert je nachdem, wie Sie die Datei im `/usr/lib/systemd/system` Verzeichnis benannt haben.

   ```
   systemctl enable trident
   ```

5. Starten Sie den Dienst.

   ```
   systemctl start trident
   ```

6. Status anzeigen.

   ```
   systemctl status trident
   ```

HINWEIS	Jedes Mal, wenn Sie die Unit-Datei ändern, führen Sie den `systemctl daemon-reload` Befehl aus, damit die Änderungen wirksam werden.

Aktualisieren oder deinstallieren Sie Trident

Sie können Trident für Docker sicher aktualisieren, ohne dass die verwendeten Volumes beeinträchtigt werden. Während des Aktualisierungsvorgangs wird es eine kurze Zeit geben, in der `docker volume` Befehle an das Plugin nicht erfolgreich sind und Anwendungen keine Volumes einbinden können, bis das Plugin wieder läuft. In den

meisten Fällen dauert dies nur wenige Sekunden.

Upgrade

Führen Sie die folgenden Schritte aus, um Trident für Docker zu aktualisieren.

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. Aktualisieren Sie das Plugin:

```
docker plugin upgrade --skip-remote-check --grant-all-permissions
netapp:latest netapp/trident-plugin:21.07
```

HINWEIS Die Version 18.01 von Trident ersetzt die nDVP. Sie sollten direkt vom `netapp/ndvp-plugin` Image auf das `netapp/trident-plugin` Image aktualisieren.

4. Aktivieren Sie das Plugin:

```
docker plugin enable netapp:latest
```

5. Überprüfen Sie, ob das Plugin aktiviert ist:

```
docker plugin ls
ID                  NAME                DESCRIPTION
ENABLED
7067f39a5df5        netapp:latest       Trident - NetApp Docker Volume
Plugin   true
```

6. Überprüfen Sie, ob die Volumes sichtbar sind:

```
docker volume ls
DRIVER              VOLUME NAME
netapp:latest       my_volume
```

WICHTIG

Wenn Sie von einer alten Version von Trident (vor 20.10) auf Trident 20.10 oder höher aktualisieren, kann ein Fehler auftreten. Weitere Informationen finden sich unter "Bekannte Probleme". Wenn Sie auf den Fehler stoßen, sollten Sie zuerst das Plugin deaktivieren, dann das Plugin entfernen und anschließend die erforderliche 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 Trident für Docker zu deinstallieren.

Schritte

1. Entfernen Sie alle Volumes, die vom Plugin erstellt wurden.
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

Sie können Volumes einfach erstellen, klonen und entfernen, indem Sie die Standard `docker volume`-Befehle mit dem angegebenen Trident-Treiber-Namen verwenden, wenn dies erforderlich ist.

Erstellen Sie ein Volumen

- Erstellen Sie ein Volume mit einem Treiber unter Verwendung des Standardnamens:

```
docker volume create -d netapp --name firstVolume
```

- Erstellen Sie ein Volume mit einer bestimmten Trident Instanz:

```
docker volume create -d ntap_bronze --name bronzeVolume
```

HINWEIS Wenn Sie keine "Optionen" angeben, werden die Standardeinstellungen des Treibers verwendet.

- Überschreiben Sie die Standardgröße des Volumes. Siehe folgendes Beispiel, um ein 20-GiB-Volume mit einem Treiber zu erstellen:

```
docker volume create -d netapp --name my_vol --opt size=20G
```

TIPP Die Volume-Größen werden als Zeichenfolgen angegeben, die einen ganzzahligen Wert mit optionalen Einheiten enthalten (Beispiel: 10G, 20GB, 3TiB). Wenn keine Einheiten angegeben sind, ist der Standard G. Größeneinheiten können entweder als Zweierpotenzen (B, KiB, MiB, GiB, TiB) oder als Zehnerpotenzen (B, KB, MB, GB, TB) angegeben werden. Kurzschreibweisen verwenden Zweierpotenzen (G = GiB, T = TiB, …).

Ein Volume entfernen

- Entfernen Sie das Volume genau wie jedes andere Docker-Volume:

```
docker volume rm firstVolume
```

WICHTIG Bei Verwendung des `solidfire-san` Treibers wird im obigen Beispiel das Volume gelöscht und bereinigt.

Führen Sie die folgenden Schritte aus, um Trident für Docker zu aktualisieren.

Ein Volume klonen

Bei Verwendung der `ontap-nas`, `ontap-san` und `solidfire-san` Speichertreiber kann Trident Volumes klonen. Bei Verwendung der `ontap-nas-flexgroup` oder `ontap-nas-economy` Treiber wird das Klonen nicht unterstützt. Das Erstellen eines neuen Volumes aus einem vorhandenen Volume führt zur Erstellung eines neuen Snapshots.

- Untersuchen Sie das Volume, um Snapshots aufzuzählen:

```
docker volume inspect <volume_name>
```

- Erstellen Sie ein neues Volume aus einem vorhandenen Volume. Dadurch wird ein neuer Snapshot erstellt:

```
docker volume create -d <driver_name> --name <new_name> -o from
=<source_docker_volume>
```

- Ein neues Volume aus einem vorhandenen Snapshot auf einem Volume erstellen. 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
```

Extern erstellte Volumes zugreifen

Auf extern erstellte Blockgeräte (oder deren Klone) können Container mit Trident **nur** zugreifen, wenn diese keine Partitionen haben und ihr Dateisystem von Trident unterstützt wird (zum Beispiel: ein `ext4`-formatiertes `/dev/sdc1` wird nicht über Trident zugänglich sein).

Treiber-spezifische Volume-Optionen

Jeder Speichertreiber verfügt über unterschiedliche Optionen, die Sie bei der Volume-Erstellung festlegen können, um das Ergebnis anzupassen. Siehe unten für Optionen, die für Ihr konfiguriertes Speichersystem gelten.

Die Verwendung dieser Optionen beim Erstellen des Volumes ist einfach. Geben Sie die Option und den Wert mit dem `-o` Operator während des CLI-Vorgangs an. Diese überschreiben alle entsprechenden Werte aus der

JSON-Konfigurationsdatei.

ONTAP Volume-Optionen

Die Optionen zum Erstellen von Volumes für NFS, iSCSI und FC umfassen Folgendes:

Option	Beschreibung
`size`	Die Größe des Volumes beträgt standardmäßig 1 GiB.
`spaceReserve`	Thin oder Thick Provisioning des Volumes, standardmäßig Thin. Gültige Werte sind `none` (thin provisioned) und `volume` (thick provisioned).
`snapshotPolicy`	Dadurch wird die Snapshot-Richtlinie auf den gewünschten Wert festgelegt. Der Standard ist `none`, was bedeutet, dass keine Snapshots automatisch für das Volume erstellt werden. Sofern Ihr Speicheradministrator diese Richtlinie nicht geändert hat, existiert auf allen ONTAP-Systemen eine Richtlinie namens „default“, die sechs stündliche, zwei tägliche und zwei wöchentliche Snapshots erstellt und speichert. Die in einem Snapshot gespeicherten Daten können wiederhergestellt werden, indem Sie zum `.snapshot`-Verzeichnis in einem beliebigen Verzeichnis des Volumes navigieren.
`snapshotReserve`	Dadurch wird die Snapshot-Reserve auf den gewünschten Prozentsatz festgelegt. Standardmäßig ist kein Wert vorhanden, was bedeutet, dass ONTAP den snapshotReserve (üblicherweise 5%) auswählt, wenn Sie eine snapshotPolicy ausgewählt haben, oder 0%, wenn die snapshotPolicy keine ist. Sie können den Standardwert für snapshotReserve in der Konfigurationsdatei für alle ONTAP Backends festlegen und ihn als Option zur Volume-Erstellung für alle ONTAP Backends außer ontap-nas-economy verwenden.
`splitOnClone`	Beim Klonen eines Volumes führt dies dazu, dass ONTAP den Klon sofort vom übergeordneten Volume trennt. Der Standardwert ist `false`. Einige Anwendungsfälle für das Klonen von Volumes werden am besten bedient, indem der Klon direkt nach der Erstellung vom übergeordneten Volume getrennt wird, da es wahrscheinlich keine Möglichkeiten für Speicheroptimierungen gibt. Zum Beispiel kann das Klonen einer leeren Datenbank viel Zeit sparen, aber nur wenig Speicherplatz, daher ist es am besten, den Klon sofort zu trennen.

Option	Beschreibung
`encryption`	Aktivieren Sie NetApp Volume Encryption (NVE) auf dem neuen Volume; Standard ist `false`. NVE muss auf dem Cluster lizenziert und aktiviert sein, um diese Option zu verwenden. Wenn NAE im Backend aktiviert ist, wird jedes in Trident bereitgestellte Volume NAE-aktiviert. Weitere Informationen finden Sie unter: "Wie Trident mit NVE und NAE zusammenarbeitet".
`tieringPolicy`	Legt die für das Volume zu verwendende Tiering-Richtlinie fest. Dies entscheidet, ob Daten in den Cloud-Tier verschoben werden, wenn sie inaktiv (kalt) werden.

Die folgenden zusätzlichen Optionen gelten **nur** für NFS:

Option	Beschreibung
`unixPermissions`	Hiermit werden die Berechtigungen für das Volume selbst festgelegt. Standardmäßig werden die Berechtigungen auf `---rwxr-xr-x` oder in numerischer Notation 0755 gesetzt, und `root` ist der Eigentümer. Sowohl das Text- als auch das Zahlenformat funktionieren.
`snapshotDir`	Wenn Sie dies auf `true` setzen, wird das `.snapshot` Verzeichnis für Clients, die auf das Volume zugreifen, sichtbar. Der Standardwert ist `false`, was bedeutet, dass die Sichtbarkeit des `.snapshot` Verzeichnisses standardmäßig deaktiviert ist. Einige Images, zum Beispiel das offizielle MySQL Image, funktionieren nicht wie erwartet, wenn das `.snapshot` Verzeichnis sichtbar ist.
`exportPolicy`	Legt die für das Volume zu verwendende Exportrichtlinie fest. Der Standardwert ist `default`.
`securityStyle`	Legt den Sicherheitsstil fest, der für den Zugriff auf das Volume verwendet wird. Der Standardwert ist `unix`. Gültige Werte sind `unix` und `mixed`.

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

Option	Beschreibung
`fileSystemType`	Legt das Dateisystem fest, das zum Formatieren von iSCSI-Volumes verwendet wird. Der Standardwert ist `ext4`. Gültige Werte sind `ext3`, `ext4` und `xfs`.
`spaceAllocation`	Wenn Sie dies auf `false` setzen, wird die Speicherplatzzuweisungsfunktion der LUN deaktiviert. Der Standardwert ist `true`, was bedeutet, dass ONTAP den Host benachrichtigt, wenn das Volume keinen Speicherplatz mehr hat und die LUN im Volume keine Schreibvorgänge mehr akzeptieren kann. Diese Option ermöglicht ONTAP außerdem, Speicherplatz automatisch freizugeben, wenn Ihr Host Daten löscht.

Beispiele

Siehe die Beispiele unten:

- Erstellen Sie ein 10-GiB-Volume:

```
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 ist und die Snapshot-Richtlinie `none` ist, verwendet Trident eine Snapshot-Reserve von 0%.

- Erstellen Sie ein Volume 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 mit einer benutzerdefinierten 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 benutzerdefinierten 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 Softwareoptionen von Element zeigen die Größe und die Quality of Service (QoS)-Richtlinien an, die dem Volume zugeordnet sind. Wenn das Volume erstellt wird, wird die zugehörige QoS-Richtlinie mit der `-o type=service_level` Nomenklatur angegeben.

Der erste Schritt zur Definition eines QoS-Dienstlevels mit dem Element-Treiber besteht darin, mindestens einen Typ zu erstellen und die minimalen, maximalen und Burst-IOPS anzugeben, die mit einem Namen in der Konfigurationsdatei verknüpft sind.

Weitere Optionen zum Erstellen von Volumes mit der Element-Software umfassen Folgendes:

Option	Beschreibung
`size`	Die Größe des Volumes, standardmäßig 1 GiB oder Konfigurationseintrag … "defaults": {"size": "5G"}.
`blocksize`	Entweder 512 oder 4096 verwenden, Standardwert ist 512 oder der Konfigurationseintrag DefaultBlockSize.

Beispiel

Siehe die folgende Beispiel-Konfigurationsdatei mit 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, Silber und Gold. Diese Namen sind willkürlich.

- Erstellen Sie ein 10 GiB Gold-Volume:

```
docker volume create -d solidfire --name sfGold -o type=Gold -o size=10G
```

- Erstelle ein 100 GiB Bronze-Volume:

```
docker volume create -d solidfire --name sfBronze -o type=Bronze -o
size=100G
```

Protokolle sammeln

Sie können Protokolle zur Unterstützung bei der Fehlerbehebung sammeln. Die Methode, mit der Sie die Protokolle sammeln, variiert je nachdem, wie Sie das Docker-Plugin ausführen.

Sammeln Sie Protokolle zur Fehlerbehebung

Schritte

1. Wenn Sie Trident mit der empfohlenen Methode zur Verwaltung von Plugins ausführen (d. h. mit `docker plugin` Befehlen), können Sie sie wie folgt anzeigen:

```
docker plugin ls
```

```
ID                  NAME                DESCRIPTION
ENABLED
4fb97d2b956b        netapp:latest       nDVP - NetApp Docker Volume
Plugin   false
journalctl -u docker | grep 4fb97d2b956b
```

   Die Standard-Protokollierungsstufe sollte Ihnen die Diagnose der meisten Probleme ermöglichen. Falls dies nicht ausreicht, können Sie die 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 die 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 die Protokolle im `/var/log/netappdvp` -Verzeichnis des Hosts verfügbar. Um die Debug-Protokollierung zu aktivieren, geben Sie `-debug` an, wenn Sie das Plugin ausführen.

Allgemeine Tipps zur Fehlerbehebung

- Das häufigste Problem, auf das neue Benutzer stoßen, ist eine Fehlkonfiguration, die die Initialisierung des Plugins verhindert. Wenn dies geschieht, wird beim Versuch, das Plugin zu installieren oder zu aktivieren, wahrscheinlich eine Meldung wie diese angezeigt:

  ```
  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. Glücklicherweise wurde das Plugin mit einer umfassenden Protokollierungsfunktion entwickelt, die Ihnen helfen sollte, die meisten Probleme zu diagnostizieren, auf die Sie wahrscheinlich stoßen werden.

- Wenn es Probleme beim Einbinden eines PV in einen Container gibt, stellen Sie sicher, dass `rpcbind` installiert und ausgeführt wird. Verwenden Sie den erforderlichen Paketmanager für das Host-Betriebssystem und prüfen Sie, ob `rpcbind` läuft. Sie können den Status des rpcbind-Dienstes überprüfen, indem Sie einen `systemctl status rpcbind` oder einen entsprechenden Befehl ausführen.

Mehrere Trident-Instanzen verwalten

Mehrere Trident-Instanzen sind erforderlich, wenn Sie mehrere Speicherkonfigurationen gleichzeitig nutzen möchten. Der Schlüssel zu mehreren Instanzen ist, ihnen unterschiedliche Namen zu geben, indem Sie die `--alias` Option mit dem containerisierten Plugin oder die `--volume-driver` Option bei der Instanziierung von Trident auf dem Host verwenden.

Schritte für das Docker-verwaltete Plugin (Version 1.13/17.03 oder später)

1. Starten Sie die erste Instanz unter Angabe eines Alias und einer Konfigurationsdatei.

   ```
   docker plugin install --grant-all-permissions --alias silver
   netapp/trident-plugin:21.07 config=silver.json
   ```

2. Starten Sie die zweite Instanz und geben Sie dabei einen anderen Alias und eine andere Konfigurationsdatei an.

   ```
   docker plugin install --grant-all-permissions --alias gold
   netapp/trident-plugin:21.07 config=gold.json
   ```

3. Erstellen Sie Volumes, indem Sie den Alias als Treibernamen angeben.

   Zum Beispiel für das Gold-Volume:

   ```
   docker volume create -d gold --name ntapGold
   ```

   Beispielsweise für Silbervolumen:

```
docker volume create -d silver --name ntapSilver
```

Schritte für traditionelle (Version 1.12 oder älter)

1. Starten Sie das Plugin 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 Plugin mit einer iSCSI-Konfiguration unter Verwendung einer benutzerdefinierten Treiber-ID:

```
sudo trident --volume-driver=netapp-san --config=/path/to/config
-iscsi.json
```

3. Docker-Volumes für jede Treiberinstanz bereitstellen:

   Zum Beispiel für NFS:

```
docker volume create -d netapp-nas --name my_nfs_vol
```

   Zum Beispiel für iSCSI:

```
docker volume create -d netapp-san --name my_iscsi_vol
```

Speicherkonfigurationsoptionen

Sehen Sie sich die für Ihre Trident Konfigurationen verfügbaren Konfigurationsoptionen an.

Globale Konfigurationsoptionen

Diese Konfigurationsoptionen gelten für alle Trident-Konfigurationen, unabhängig von der verwendeten Storage-Plattform.

Option	Beschreibung	Beispiel
`version`	Versionsnummer der Konfigurationsdatei	`1`

Option	Beschreibung	Beispiel
`storageDriverName`	Name des Speichertreibers	`ontap-nas`, `ontap-san`, `ontap-nas-economy`, `ontap-nas-flexgroup`, `solidfire-san`
`storagePrefix`	Optionales Präfix für Datenträgernamen. Standard: `netappdvp_`.	`staging_`
`limitVolumeSize`	Optionale Beschränkung der Volumengrößen. Standard: "" (nicht erzwungen)	`10g`

TIPP

Verwenden Sie `storagePrefix` (einschließlich der Standardeinstellung) nicht für Element-Backends. Standardmäßig ignoriert der `solidfire-san` Treiber diese Einstellung und verwendet kein Präfix. NetApp empfiehlt, entweder eine spezifische tenantID für das Docker-Volume-Mapping zu verwenden oder die Attributdaten zu nutzen, die mit der Docker-Version, Treiberinformationen und dem Rohnamen von Docker gefüllt sind, falls eine Namensmanipulation vorgenommen wurde.

Standardoptionen sind verfügbar, sodass Sie diese nicht für jedes erstellte Volume angeben müssen. Die `size` Option steht für alle Controller-Typen zur Verfügung. Siehe den Abschnitt zur ONTAP-Konfiguration für ein Beispiel, wie die Standard-Volume-Größe festgelegt wird.

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 die folgenden 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 Protokoll-LIF. **ONTAP NAS-Treiber**: NetApp empfiehlt, `dataLIF` anzugeben. Falls nicht angegeben, ruft Trident dataLIFs vom SVM ab. Ein vollqualifizierter Domainname (FQDN) kann für die NFS-Mount-Operationen angegeben werden, wodurch die Erstellung eines Round-Robin-DNS zur Lastverteilung über mehrere dataLIFs ermöglicht wird. **ONTAP SAN drivers**: Nicht für iSCSI oder FC angeben. Trident verwendet "ONTAP Selective LUN Map" zur Erkennung der iSCSI- oder FC-LIFs, die für die Einrichtung einer Multipath-Sitzung benötigt werden. Eine Warnung wird generiert, wenn `dataLIF` explizit definiert ist.	`10.0.0.2`
`svm`	Zu verwendende Storage Virtual Machine (erforderlich, wenn die Management-LIF eine Cluster-LIF ist)	`svm_nfs`
`username`	Benutzername für die Verbindung zum Speichergerät	`vsadmin`
`password`	Passwort zum Verbinden mit dem Speichergerät	`secret`
`aggregate`	Aggregat für die Bereitstellung (optional; falls festgelegt, muss es der SVM zugewiesen werden). Für den `ontap-nas-flexgroup` Treiber wird diese Option ignoriert. Alle Aggregate, die der SVM zugewiesen sind, werden zur Bereitstellung eines FlexGroup-Volumes verwendet.	`aggr1`
`limitAggregateUsage`	Optional, die Bereitstellung schlägt fehl, wenn die Nutzung über diesem Prozentsatz liegt	`75%`

Option	Beschreibung	Beispiel
`nfsMountOptions`	Feingranulare Steuerung der NFS-Mount-Optionen; Standardwert ist "-o nfsvers=3". **Nur für die `ontap-nas` und `ontap-nas-economy` Treiber verfügbar**. "Informationen zur NFS-Hostkonfiguration finden Sie hier".	`-o nfsvers=4`
`igroupName`	Trident erstellt und verwaltet pro Node `igroups` als `netappdvp`. Dieser Wert darf weder geändert noch weggelassen werden. **Nur für den `ontap-san` Treiber verfügbar**.	`netappdvp`
`limitVolumeSize`	Maximal anforderbare Volume-Größe.	`300g`
`qtreesPerFlexvol`	Maximale Qtrees pro FlexVol, muss im Bereich [50, 300] liegen, Standard ist 200. **Für den `ontap-nas-economy` driver ermöglicht diese Option die Anpassung der maximalen Anzahl von qtrees pro FlexVol**.	`300`
`sanType`	**Unterstützt nur für `ontap-san` -Treiber.** Verwenden Sie zum Auswählen `iscsi` für iSCSI, `nvme` für NVMe/TCP oder `fcp` für SCSI über Fibre Channel (FC).	`iscsi` falls leer
`limitVolumePoolSize`	**Unterstützt nur für `ontap-san-economy` und `ontap-san-economy` Treiber.** Begrenzt FlexVol-Größen in ONTAP ontap-nas-economy und ontap-SAN-economy Treibern.	`300g`

Standardoptionen sind verfügbar, damit Sie diese nicht bei jedem erstellten Volume angeben müssen:

Option	Beschreibung	Beispiel
`spaceReserve`	Speicherplatzreservierungsmodus; `none` (dünn bereitgestellt) oder `volume` (dick)	`none`

Option	Beschreibung	Beispiel
`snapshotPolicy`	Zu verwendende Snapshot-Richtlinie, Standard ist `none`	`none`
`snapshotReserve`	Snapshot-Reserveprozentsatz, Standardwert ist "", um den ONTAP-Standardwert zu akzeptieren	`10`
`splitOnClone`	Einen Klon beim Erstellen von seinem übergeordneten Objekt trennen, Standard ist `false`	`false`
`encryption`	Aktiviert NetApp Volume Encryption (NVE) auf dem neuen Volume; standardmäßig `false`. NVE muss auf dem Cluster lizenziert und aktiviert sein, um diese Option zu verwenden. Wenn NAE im Backend aktiviert ist, wird jedes in Trident bereitgestellte Volume NAE-aktiviert. Weitere Informationen finden Sie unter: "Wie Trident mit NVE und NAE zusammenarbeitet".	true
`unixPermissions`	NAS-Option für bereitgestellte NFS-Volumes, Standardwert ist `777`	`777`
`snapshotDir`	NAS-Option für den Zugriff auf das `.snapshot` Verzeichnis.	Aktiviert oder deaktiviert den Zugriff auf das `.snapshot` Verzeichnis. Gültige Werte: `true`, `false`. Das Standardverhalten kann je nach ONTAP und Trident Version variieren. Explizit festlegen, um ein konsistentes Verhalten sicherzustellen.
`exportPolicy`	NAS-Option für die zu verwendende NFS-Exportrichtlinie, Standardwert ist `default`	`default`
`securityStyle`	NAS option für den Zugriff auf das bereitgestellte NFS-Volume. NFS unterstützt `mixed` und `unix` Sicherheitsstile. Der Standardwert ist `unix`.	`unix`
`fileSystemType`	SAN-Option zur Auswahl des Dateisystemtyps, Standardwert ist `ext4`	`xfs`
`tieringPolicy`	Zu verwendende Tiering-Richtlinie, Standard ist `none`.	`none`

Option	Beschreibung	Beispiel
`skipRecovery Queue`	Während des Löschens eines Volumes wird die Wiederherstellungswarteschlange im Speicher umgangen und das Volume sofort gelöscht.	``

Skalierungsoptionen

Die `ontap-nas` und `ontap-san` Treiber erstellen für jedes Docker-Volume ein ONTAP FlexVol. ONTAP unterstützt bis zu 1000 FlexVols pro Clusterknoten mit einem Cluster-Maximum von 12.000 FlexVol Volumes. Wenn Ihre Docker-Volume-Anforderungen innerhalb dieser Beschränkung liegen, ist der `ontap-nas` Treiber die bevorzugte NAS-Lösung aufgrund der zusätzlichen Funktionen, die FlexVols bieten, wie Docker-Volume-genaue Snapshots und Klonen.

Falls Sie mehr Docker-Volumes benötigen, als die FlexVol-Beschränkungen zulassen, wählen Sie den `ontap-nas-economy` oder den `ontap-san-economy` Treiber.

Der `ontap-nas-economy` Treiber erstellt Docker-Volumes als ONTAP Qtrees innerhalb eines Pools von automatisch verwalteten FlexVol Volumes. Qtrees bieten eine deutlich höhere Skalierbarkeit, bis zu 100.000 pro Clusterknoten und 2.400.000 pro Cluster, allerdings auf Kosten einiger Funktionen. Der `ontap-nas-economy` Treiber unterstützt keine Docker-Volume-granularen Snapshots oder das Klonen.

HINWEIS	Der `ontap-nas-economy` Treiber wird derzeit in Docker Swarm nicht unterstützt, da Docker Swarm die Erstellung von Volumes nicht über mehrere Knoten hinweg orchestriert.

Der `ontap-san-economy` Treiber erstellt Docker-Volumes als ONTAP-LUNs innerhalb eines gemeinsam genutzten Pools von automatisch verwalteten FlexVol Volumes. Auf diese Weise ist jedes FlexVol nicht auf nur eine LUN beschränkt und bietet eine bessere Skalierbarkeit für SAN-Workloads. Abhängig vom Storage-Array unterstützt ONTAP bis zu 16384 LUNs pro Cluster. Da die Volumes LUNs zugrunde liegen, unterstützt dieser Treiber Docker-Volume-genaue Snapshots und das Klonen.

Wählen Sie den `ontap-nas-flexgroup` Treiber, um die Parallelität auf ein einzelnes Volume zu erhöhen, das in den Petabyte-Bereich mit Milliarden von Dateien wachsen kann. Einige ideale Anwendungsfälle für FlexGroups sind KI/ML/DL, Big Data und Analysen, Software-Builds, Streaming, Datei-Repositorien und so weiter. Trident verwendet alle Aggregate, die einer SVM zugewiesen sind, wenn ein FlexGroup Volume bereitgestellt wird. FlexGroup Unterstützung in Trident hat außerdem folgende Überlegungen:

- Erfordert ONTAP Version 9.2 oder höher.
- Zum jetzigen Zeitpunkt unterstützen FlexGroups nur NFS v3.
- Es wird empfohlen, die 64-Bit-NFSv3-Kennungen für die SVM zu aktivieren.
- Die empfohlene Mindestgröße eines FlexGroup-Mitglieds/Volumes beträgt 100 GiB.
- Das Klonen von FlexGroup-Volumes wird nicht unterstützt.

Weitere Informationen zu FlexGroups und Arbeitslasten, die für FlexGroups geeignet sind, finden Sie im "NetApp FlexGroup Volume Best Practices und Implementierungsleitfaden".

Um erweiterte Funktionen und große Skalierbarkeit in derselben Umgebung zu erreichen, können Sie mehrere Instanzen des Docker Volume Plugins ausführen, wobei eine `ontap-nas` und eine andere `ontap-nas-economy` verwendet.

Benutzerdefinierte ONTAP-Rolle für Trident

Sie können eine ONTAP-Clusterrolle mit minimalen Berechtigungen erstellen, sodass Sie nicht die ONTAP-Admin-Rolle verwenden müssen, um Vorgänge in Trident durchzuführen. Wenn Sie den Benutzernamen in einer Trident-Backend-Konfiguration angeben, verwendet Trident die von Ihnen erstellte ONTAP-Clusterrolle, um die Vorgänge auszuführen.

Weitere Informationen zum Erstellen benutzerdefinierter Trident-Rollen finden Sie unter "Trident Custom-Role-Generator".

Verwendung der ONTAP-Befehlszeile

1. Erstellen Sie eine neue Rolle mit folgendem Befehl:

```
security login role create <role_name\> -cmddirname "command" -access all
-vserver <svm_name\>
```

2. Erstellen Sie einen Benutzernamen für den Trident Benutzer:

```
security login create -username <user_name\> -application ontapi
-authmethod password -role <name_of_role_in_step_1\> -vserver <svm_name\>
-comment "user_description"
security login create -username <user_name\> -application http -authmethod
password -role <name_of_role_in_step_1\> -vserver <svm_name\> -comment
"user_description"
```

3. Ordnen Sie die Rolle dem Benutzer zu:

```
security login modify username <user_name\> -vserver <svm_name\> -role
<role_name\> -application ontapi -application console -authmethod
<password\>
```

Verwenden von System Manager

Führen Sie die folgenden Schritte im ONTAP System Manager aus:

1. **Erstellen Sie eine benutzerdefinierte Rolle**:
   a. Um eine benutzerdefinierte Rolle auf Clusterebene zu erstellen, wählen Sie **Cluster > Settings**.

      (Oder) Um eine benutzerdefinierte Rolle auf SVM-Ebene zu erstellen, wählen Sie **Storage > Storage VMs > `required SVM`> Settings > Users and Roles**.

   b. Wählen Sie das Pfeilsymbol (→) neben **Users and Roles** aus.
   c. Wählen Sie unter **Rollen +Add** aus.
   d. Definieren Sie die Regeln für die Rolle und klicken Sie auf **Save**.
2. **Rolle dem Trident-Benutzer zuordnen**: + Führen Sie die folgenden Schritte auf der Seite **Benutzer und Rollen** aus:
   a. Wählen Sie das Symbol **+** unter **Benutzer** aus.
   b. Wählen Sie den gewünschten Benutzernamen aus und wählen Sie eine Rolle im Dropdown-Menü für **Rolle**.
   c. Klicken Sie auf **Speichern**.

Weitere Informationen finden Sie auf den folgenden Seiten:

- "Benutzerdefinierte Rollen für die Administration von ONTAP" oder "Benutzerdefinierte Rollen definieren"
- "Mit Rollen und Benutzern arbeiten"

Beispiel ONTAP-Konfigurationsdateien

NFS-Beispiel für <code>ontap-nas</code> Treiber

```
{
  "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 <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": {
    "size": "100G",
    "spaceReserve": "none",
    "exportPolicy": "default"
  }
}
```

NFS-Beispiel für <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",
  "aggregate": "aggr1"
}
```

iSCSI-Beispiel für <code>ontap-san</code>-Treiber

```
{
  "version": 1,
  "storageDriverName": "ontap-san",
  "managementLIF": "10.0.0.1",
  "dataLIF": "10.0.0.3",
  "svm": "svm_iscsi",
  "username": "vsadmin",
  "password": "password",
  "aggregate": "aggr1",
  "igroupName": "netappdvp"
}
```

NFS-Beispiel für <code>ontap-san-economy</code>-Treiber

```
{
  "version": 1,
  "storageDriverName": "ontap-san-economy",
  "managementLIF": "10.0.0.1",
  "dataLIF": "10.0.0.3",
  "svm": "svm_iscsi_eco",
  "username": "vsadmin",
  "password": "password",
  "aggregate": "aggr1",
  "igroupName": "netappdvp"
}
```

NVMe/TCP-Beispiel für <code>ontap-san</code>-Treiber

```
{
  "version": 1,
  "backendName": "NVMeBackend",
  "storageDriverName": "ontap-san",
  "managementLIF": "10.0.0.1",
  "svm": "svm_nvme",
  "username": "vsadmin",
  "password": "password",
  "sanType": "nvme",
  "useREST": true
}
```

SCSI über FC Beispiel für <code>ontap-san</code> Treiber

```
{
  "version": 1,
  "backendName": "ontap-san-backend",
  "storageDriverName": "ontap-san",
  "managementLIF": "10.0.0.1",
  "sanType": "fcp",
  "svm": "trident_svm",
  "username": "vsadmin",
  "password": "password",
  "useREST": true
}
```

Element-Softwarekonfiguration

Zusätzlich zu den globalen Konfigurationswerten stehen bei Verwendung der Element-Software (NetApp HCI/SolidFire) diese Optionen zur Verfügung.

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
`TenantName`	SolidFireF Tenant to use (wird erstellt, falls nicht gefunden)	`docker`

Option	Beschreibung	Beispiel
`InitiatorIFace`	Schnittstelle angeben, wenn der iSCSI-Datenverkehr auf eine nicht standardmäßige Schnittstelle beschränkt wird	`default`
`Types`	QoS-Spezifikationen	Siehe Beispiel unten
`LegacyNamePrefix`	Präfix für aktualisierte Trident-Installationen. Wenn Sie eine Version von Trident vor 1.3.2 verwendet haben und ein Upgrade mit bestehenden 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` Treiber unterstützt Docker Swarm nicht.

Beispiel einer Element-Konfigurationsdatei

```
{
  "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 Trident mit Docker.

Das Upgrade des Trident Docker Volume Plugin auf 20.10 und höher von älteren Versionen führt zu einem Upgrade-Fehler mit der Fehlermeldung no such file or directory.

Problemumgehung

1. Deaktivieren Sie das Plugin.

```
docker plugin disable -f netapp:latest
```

2. Entfernen Sie das Plugin.

```
docker plugin rm -f netapp:latest
```

3. Installieren Sie das Plugin erneut, indem Sie den zusätzlichen `config` Parameter angeben.

```
docker plugin install netapp/trident-plugin:20.10 --alias netapp --grant
-all-permissions config=config.json
```

Volume-Namen müssen mindestens 2 Zeichen lang sein.

HINWEIS	Dies ist eine Einschränkung des Docker-Clients. Der Client interpretiert einen einstelligen Namen als Windows-Pfad "Siehe Bug 25773".

Docker Swarm weist bestimmte Verhaltensweisen auf, die Trident daran hindern, es mit jeder Speicher- und Treiberkombination zu unterstützen.

- Docker Swarm verwendet derzeit den Volumennamen anstelle der Volume-ID als eindeutigen Volume-Bezeichner.
- Volumenanforderungen werden gleichzeitig an jeden Knoten in einem Swarm-Cluster gesendet.
- Volume-Plugins (einschließlich Trident) müssen auf jedem Knoten eines Swarm-Clusters unabhängig ausgeführt werden. Aufgrund der Funktionsweise von ONTAP und wie die `ontap-nas` und `ontap-san` Treiber funktionieren, sind sie die einzigen, die in der Lage sind, innerhalb dieser Einschränkungen zu arbeiten.

Die übrigen Treiber sind Problemen wie Race Conditions ausgesetzt, die dazu führen können, dass für eine einzelne Anfrage eine große Anzahl von Volumes erstellt wird, ohne dass es einen klaren "Gewinner" gibt; beispielsweise verfügt Element über eine Funktion, die es ermöglicht, dass Volumes den gleichen Namen, aber unterschiedliche IDs haben.

NetApp has dem Docker-Team Feedback gegeben, aber es gibt keine Hinweise auf zukünftige Maßnahmen.

Wenn ein FlexGroup bereitgestellt wird, stellt ONTAP kein zweites FlexGroup bereit, wenn das zweite FlexGroup ein oder mehrere Aggregate mit dem FlexGroup, das bereitgestellt wird, gemeinsam hat.

Bewährte Verfahren und Empfehlungen

Bereitstellung

Verwenden Sie die hier aufgeführten Empfehlungen, wenn Sie Trident bereitstellen.

In einem dedizierten Namensraum bereitstellen

"Namensräume"Sie sorgen für eine administrative Trennung zwischen verschiedenen Anwendungen und stellen eine Barriere für die gemeinsame Nutzung von Ressourcen dar. Beispielsweise kann ein PVC aus einem Namespace nicht von einem anderen verwendet werden. Trident stellt PV-Ressourcen allen Namespaces im Kubernetes-Cluster zur Verfügung und nutzt daher ein Servicekonto mit erweiterten Berechtigungen.

Darüber hinaus könnte der Zugriff auf den Trident-Pod einem Benutzer den Zugriff auf Anmeldeinformationen des Speichersystems und andere sensible Informationen ermöglichen. Es ist wichtig sicherzustellen, dass Anwendungsbenutzer und Verwaltungsanwendungen nicht die Möglichkeit haben, auf die Trident-Objektdefinitionen oder die Pods selbst zuzugreifen.

Verwenden Sie Quoten und Bereichsgrenzen, um den Speicherverbrauch zu kontrollieren

Kubernetes verfügt über zwei Funktionen, die in Kombination einen leistungsstarken Mechanismus zur Begrenzung des Ressourcenverbrauchs von Anwendungen bieten. Der "Speicherquotenmechanismus" ermöglicht es dem Administrator, globale sowie speicherklassenspezifische Kapazitäts- und Objektanzahlbeschränkungen pro Namespace zu implementieren. Darüber hinaus stellt die Verwendung eines "Bereichslimit" sicher, dass die PVC-Anfragen sowohl einen minimalen als auch einen maximalen Wert einhalten, bevor die Anfrage an den Provisioner weitergeleitet wird.

Diese Werte werden pro Namensraum definiert, was bedeutet, dass für jeden Namensraum Werte definiert sein sollten, die seinen Ressourcenanforderungen entsprechen. Siehe hier für Informationen über "wie man Quoten nutzt".

Storage-Konfiguration

Jede Speicherplattform im NetApp Portfolio verfügt über einzigartige Funktionen, die Anwendungen zugutekommen, ob containerisiert oder nicht.

Plattformübersicht

Trident ist mit ONTAP und Element kompatibel. Es gibt keine Plattform, die für alle Anwendungen und Szenarien besser geeignet ist als eine andere, jedoch sollten bei der Auswahl einer Plattform die Anforderungen der Anwendung und des Teams, das das Gerät verwaltet, berücksichtigt werden.

Sie sollten die grundlegenden Best Practices für das Host-Betriebssystem mit dem Protokoll, das Sie verwenden, beachten. Optional können Sie in Erwägung ziehen, sofern verfügbar, die Best Practices für die Anwendung zusammen mit Backend-, Speicherklassen- und PVC-Einstellungen zu integrieren, um den Speicher für spezifische Anwendungen zu optimieren.

ONTAP und Cloud Volumes ONTAP Best Practices

Lernen Sie die Best Practices für die Konfiguration von ONTAP und Cloud Volumes ONTAP für Trident.

Die folgenden Empfehlungen sind Richtlinien für die Konfiguration von ONTAP für containerisierte Workloads, die Volumes nutzen, die von Trident dynamisch bereitgestellt werden. Jede sollte hinsichtlich ihrer Eignung für Ihre Umgebung geprüft und bewertet werden.

Verwenden Sie SVM(s), die speziell für Trident vorgesehen sind

Storage Virtual Machines (SVMs) bieten Isolation und administrative Trennung zwischen Mandanten auf einem ONTAP-System. Die Zuweisung einer SVM zu Anwendungen ermöglicht die Delegation von Berechtigungen und die Anwendung von Best Practices zur Begrenzung des Ressourcenverbrauchs.

Für die Verwaltung der SVM stehen mehrere Optionen zur Verfügung:

- Stellen Sie die Cluster-Managementoberfläche in der Backend-Konfiguration zusammen mit den entsprechenden Anmeldeinformationen bereit und geben Sie den SVM-Namen an.
- Erstellen Sie eine dedizierte Managementoberfläche für die SVM mithilfe von ONTAP System Manager oder der CLI.
- Teilen Sie die Verwaltungsrolle mit einer NFS-Datenschnittstelle.

In jedem Fall sollte die Schnittstelle im DNS registriert sein, und der DNS-Name sollte bei der Konfiguration von Trident verwendet werden. Dies erleichtert einige DR-Szenarien, zum Beispiel SVM-DR ohne die Verwendung der Netzwerkidentitätsbeibehaltung.

Es gibt keine Präferenz zwischen einer dedizierten oder gemeinsam genutzten Management-LIF für die SVM, jedoch sollten Sie sicherstellen, dass Ihre Netzwerksicherheitsrichtlinien mit dem von Ihnen gewählten Ansatz übereinstimmen. Unabhängig davon sollte die Management-LIF über DNS erreichbar sein, um maximale Flexibilität zu ermöglichen, falls "SVM-DR" in Verbindung mit Trident verwendet wird.

Begrenzen Sie die maximale Volume-Anzahl

ONTAP-Speichersysteme haben eine maximale Volume-Anzahl, die je nach Softwareversion und Hardwareplattform variiert. Siehe "NetApp Hardware Universe" für Ihre spezifische Plattform und ONTAP-Version, um die genauen Grenzwerte zu bestimmen. Wenn die Volume-Anzahl erschöpft ist, schlagen Bereitstellungsvorgänge nicht nur für Trident, sondern für alle Speicheranforderungen fehl.

Tridents `ontap-nas` und `ontap-san` Treiber stellen ein FlexVolume für jedes erstellte Kubernetes Persistent Volume (PV) bereit. Der `ontap-nas-economy` Treiber erstellt etwa ein FlexVolume für alle 200 PVs (konfigurierbar zwischen 50 und 300). Der `ontap-san-economy` Treiber erstellt etwa ein FlexVolume für alle 100 PVs (konfigurierbar zwischen 50 und 200). Um zu verhindern, dass Trident alle verfügbaren Volumes auf dem Speichersystem verbraucht, sollten Sie ein Limit für die SVM festlegen. Sie können dies über die Befehlszeile tun:

```
vserver modify -vserver <svm_name> -max-volumes <num_of_volumes>
```

Der Wert für `max-volumes` variiert je nach mehreren Kriterien, die für Ihre Umgebung spezifisch sind:

- Die Anzahl der vorhandenen Volumes im ONTAP-Cluster
- Die Anzahl der Volumes, die Sie voraussichtlich außerhalb von Trident für andere Anwendungen

bereitstellen werden

- Die Anzahl der persistenten Volumes, die voraussichtlich von Kubernetes-Anwendungen verwendet werden

Der `max-volumes` Wert ist die Gesamtanzahl der über alle Nodes im ONTAP Cluster bereitgestellten Volumes und nicht auf einem einzelnen ONTAP Node. Daher kann es vorkommen, dass ein ONTAP Cluster-Node deutlich mehr oder weniger von Trident bereitgestellte Volumes hat als ein anderer Node.

Ein ONTAP-Cluster mit zwei Knoten kann beispielsweise maximal 2000 FlexVol Volumes hosten. Die Festlegung der maximalen Volume-Anzahl auf 1250 erscheint sehr sinnvoll. Wenn jedoch nur "Aggregate" von einem Knoten dem SVM zugewiesen sind oder die Aggregate, die von einem Knoten zugewiesen wurden, nicht bereitgestellt werden können (zum Beispiel aufgrund von Kapazität), wird der andere Knoten zum Ziel für alle von Trident bereitgestellten Volumes. Das bedeutet, dass das Volume-Limit für diesen Knoten möglicherweise erreicht wird, bevor der `max-volumes` Wert erreicht ist, was sowohl Trident als auch andere Volume-Operationen, die diesen Knoten nutzen, beeinträchtigt. **Sie können diese Situation vermeiden, indem Sie sicherstellen, dass Aggregate von jedem Knoten im Cluster dem von Trident verwendeten SVM in gleicher Anzahl zugewiesen werden.**

Ein Volume klonen

NetApp Trident unterstützt das Klonen von Volumes bei Verwendung der `ontap-nas`, `ontap-san` und `solidfire-san` Speichertreiber. Bei Verwendung der `ontap-nas-flexgroup` oder `ontap-nas-economy` Treiber wird das Klonen nicht unterstützt. Das Erstellen eines neuen Volumes aus einem vorhandenen Volume führt zur Erstellung eines neuen Snapshots.

WARNUNG	Vermeiden Sie das Klonen einer PVC, die mit einer anderen StorageClass verknüpft ist. Führen Sie Klonvorgänge innerhalb derselben StorageClass durch, um Kompatibilität zu gewährleisten und unerwartetes Verhalten zu verhindern.

Begrenzen Sie die maximale Größe von Volumes, die von Trident erstellt werden

Um die maximale Größe für Volumes zu konfigurieren, die von Trident erstellt werden können, verwenden Sie den `limitVolumeSize` Parameter in Ihrer `backend.json` Definition.

Neben der Kontrolle der Volume-Größe am Speicherarray sollten Sie auch die Kubernetes-Funktionen nutzen.

Begrenzen Sie die maximale Größe der FlexVols, die von Trident erstellt werden

Um die maximale Größe für FlexVols, die als Pools für ontap-san-economy und ontap-nas-economy Treiber verwendet werden, zu konfigurieren, verwenden Sie den `limitVolumePoolSize` Parameter in Ihrer `backend.json` Definition.

Konfigurieren Sie Trident für die Verwendung von bidirektionalem CHAP

Sie können die CHAP-Initiator- und Ziel-Benutzernamen sowie Passwörter in Ihrer Backend-Definition angeben und Trident CHAP auf der SVM aktivieren lassen. Mit dem `useCHAP`-Parameter in Ihrer Backend-Konfiguration authentifiziert Trident iSCSI-Verbindungen für ONTAP-Backends mit CHAP.

Erstellen und Verwenden einer SVM-QoS-Richtlinie

Durch die Nutzung einer ONTAP-QoS-Richtlinie, die auf die SVM angewendet wird, wird die Anzahl der von den durch Trident bereitgestellten Volumes verbrauchbaren IOPS begrenzt. Dies hilft, "einen Tyrannen verhindern"dass ein außer Kontrolle geratener Container keine Workloads außerhalb der Trident SVM

beeinträchtigt.

Sie können in wenigen Schritten eine QoS-Richtlinie für die SVM erstellen. Siehe die Dokumentation für Ihre Version von ONTAP für die genauesten Informationen. Das folgende Beispiel erstellt eine QoS-Richtlinie, die die insgesamt für die 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>
```

Zusätzlich, wenn Ihre Version von ONTAP dies unterstützt, können Sie die Verwendung eines QoS-Minimums in Betracht ziehen, um einen bestimmten Durchsatz für containerisierte Workloads zu garantieren. Adaptives QoS ist nicht mit einer SVM-Level-Richtlinie kompatibel.

Die Anzahl der für die containerisierten Workloads reservierten IOPS hängt von vielen Aspekten ab. Dazu gehören unter anderem:

- Andere Workloads, die das Speichersystem nutzen. Falls andere Workloads, die nicht mit der Kubernetes-Bereitstellung zusammenhängen, die Speicherressourcen nutzen, ist darauf zu achten, dass diese Workloads nicht versehentlich negativ beeinflusst werden.
- Erwartete Workloads, die in Containern ausgeführt werden. Wenn Workloads mit hohen IOPS-Anforderungen in Containern ausgeführt werden, führt eine niedrige QoS-Richtlinie zu einer schlechten Benutzererfahrung.

Es ist wichtig zu beachten, dass eine auf SVM-Ebene zugewiesene QoS-Richtlinie dazu führt, dass alle für die SVM bereitgestellten Volumes denselben IOPS-Pool nutzen. Wenn eine oder wenige der containerisierten Anwendungen einen hohen IOPS-Bedarf haben, könnten sie zu einem „Bully“ für die anderen containerisierten Workloads werden. Wenn dies der Fall ist, sollten Sie in Erwägung ziehen, externe Automatisierung zu verwenden, um QoS-Richtlinien pro Volume zuzuweisen.

WICHTIG	Sie sollten die QoS-Richtliniengruppe der SVM **nur** zuweisen, wenn Ihre ONTAP Version älter als 9.8 ist.

QoS-Policy-Gruppen für Trident erstellen

Quality of Service (QoS) gewährleistet, dass die Leistung kritischer Workloads nicht durch konkurrierende Workloads beeinträchtigt wird. ONTAP QoS-Richtliniengruppen bieten QoS-Optionen für Volumes und ermöglichen Benutzern, die Durchsatzobergrenze für einen oder mehrere Workloads festzulegen. Weitere Informationen zu QoS finden Sie unter "Gewährleistung des Durchsatzes mit QoS". Sie können QoS-Richtliniengruppen im Backend oder in einem Speicherpool angeben, und sie werden auf jedes in diesem Pool oder Backend erstellte Volume angewendet.

ONTAP bietet zwei Arten von QoS-Richtliniengruppen: traditionelle und adaptive. Traditionelle Richtliniengruppen bieten einen festen maximalen (oder in späteren Versionen minimalen) Durchsatz in IOPS. Adaptives QoS skaliert den Durchsatz automatisch mit der Workload-Größe und hält das Verhältnis von IOPS zu TB/GB aufrecht, wenn sich die Größe der Workload ändert. Dies bietet einen erheblichen Vorteil, wenn Sie Hunderte oder Tausende von Workloads in einer großen Umgebung verwalten.

Beachten Sie Folgendes, wenn Sie QoS-Richtliniengruppen erstellen:

- Sie sollten den `qosPolicy` Schlüssel im `defaults` Block der Backend-Konfiguration festlegen. Siehe das folgende Beispiel für eine Backend-Konfiguration:

```
---
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, sodass jedes Volume den gesamten Durchsatz erhält, wie von der Richtliniengruppe festgelegt. Gemeinsam genutzte Richtliniengruppen werden nicht unterstützt.

Weitere Informationen zu QoS-Richtliniengruppen finden Sie unter "ONTAP-Befehlsreferenz".

Beschränken Sie den Zugriff auf Speicherressourcen auf Mitglieder des Kubernetes-Clusters

Die Beschränkung des Zugriffs auf die von Trident erstellten NFS-Volumes, iSCSI-LUNs und FC-LUNs ist ein entscheidender Bestandteil der Sicherheitsarchitektur Ihrer Kubernetes-Bereitstellung. Dadurch wird verhindert, dass Hosts, die nicht Teil des Kubernetes-Clusters sind, auf die Volumes zugreifen und möglicherweise Daten unerwartet verändern.

Es ist wichtig zu verstehen, dass Namespaces die logische Grenze für Ressourcen in Kubernetes darstellen. Die Annahme ist, dass Ressourcen im selben Namespace gemeinsam genutzt werden können, jedoch gibt es, was wichtig ist, keine Namespace-übergreifende Fähigkeit. Das bedeutet, dass selbst wenn PVs globale Objekte sind, sie, wenn sie an ein PVC gebunden sind, nur von Pods im selben Namespace zugänglich sind. **Es ist entscheidend sicherzustellen, dass Namespaces verwendet werden, um bei Bedarf eine Trennung bereitzustellen.**

Die größte Sorge der meisten Organisationen hinsichtlich Datensicherheit im Kubernetes-Kontext besteht darin, dass ein Prozess in einem Container auf Speicher zugreifen kann, der auf dem Host eingebunden ist, aber nicht für den Container bestimmt ist. "Namensräume" sind darauf ausgelegt, diese Art von Kompromittierung zu verhindern. Es gibt jedoch eine Ausnahme: privilegierte Container.

Ein privilegierter Container ist ein Container, der mit deutlich mehr Host-Berechtigungen als üblich ausgeführt wird. Diese werden nicht standardmäßig verweigert, daher stellen Sie sicher, dass Sie die Fähigkeit mit "Sicherheitsrichtlinien für Pods" deaktivieren.

Für Volumes, auf die sowohl von Kubernetes als auch von externen Hosts zugegriffen werden soll, sollte der Speicher auf traditionelle Weise verwaltet werden, wobei das PV vom Administrator eingeführt und nicht von Trident verwaltet wird. Dies stellt sicher, dass das Speichervolume nur dann zerstört wird, wenn sowohl Kubernetes als auch die externen Hosts die Verbindung getrennt haben und das Volume nicht mehr verwenden. Zusätzlich kann eine benutzerdefinierte Exportrichtlinie angewendet werden, die den Zugriff von den Kubernetes-Clusterknoten und von Zielservern außerhalb des Kubernetes-Clusters ermöglicht.

Für Bereitstellungen mit dedizierten Infrastrukturknoten (zum Beispiel OpenShift) oder anderen Knoten, die keine Benutzeranwendungen planen können, sollten separate Exportrichtlinien verwendet werden, um den Zugriff auf Speicherressourcen weiter einzuschränken. Dies umfasst die Erstellung einer Exportrichtlinie für Dienste, die auf diesen Infrastrukturknoten bereitgestellt werden (zum Beispiel die OpenShift Metrics- und Logging-Dienste), sowie für Standardanwendungen, die auf Nicht-Infrastrukturknoten bereitgestellt werden.

Verwenden Sie eine dedizierte Exportrichtlinie

Sie sollten sicherstellen, dass für jedes Backend eine Exportrichtlinie existiert, die den Zugriff ausschließlich auf die im Kubernetes-Cluster vorhandenen Knoten erlaubt. Trident kann Exportrichtlinien automatisch erstellen und verwalten. Auf diese Weise beschränkt Trident den Zugriff auf die von ihm bereitgestellten Volumes auf die Knoten im Kubernetes-Cluster und vereinfacht das Hinzufügen und Entfernen von Knoten.

Alternativ können Sie auch manuell eine Exportrichtlinie erstellen und sie mit einer oder mehreren Exportregeln versehen, die jede Knotenzugriffsanfrage verarbeiten:

- Verwenden Sie den `vserver export-policy create` ONTAP CLI-Befehl, um die Export-Policy zu erstellen.
- Fügen Sie der Exportrichtlinie Regeln mithilfe des `vserver export-policy rule create` ONTAP CLI-Befehls hinzu.

Durch das Ausführen dieser Befehle können Sie einschränken, welche Kubernetes-Knoten Zugriff auf die Daten haben.

Deaktivieren `showmount` für die Anwendung SVM

Die `showmount` Funktion ermöglicht es einem NFS-Client, die SVM nach einer Liste verfügbarer NFS-Exporte abzufragen. Ein im Kubernetes-Cluster bereitgestellter Pod kann den `showmount -e` Befehl gegen die SVM ausführen und eine Liste der verfügbaren Mounts erhalten, einschließlich solcher, auf die er keinen Zugriff hat. Auch wenn dies für sich genommen kein Sicherheitsrisiko darstellt, liefert es dennoch unnötige Informationen, die einem unbefugten Benutzer potenziell helfen können, eine Verbindung zu einem NFS-Export herzustellen.

Sie sollten `showmount` mithilfe des ONTAP CLI-Befehls auf SVM-Ebene deaktivieren:

```
vserver nfs modify -vserver <svm_name> -showmount disabled
```

SolidFire Best Practices

Lernen Sie die Best Practices für die Konfiguration von SolidFire-Speicher für Trident.

SolidFire-Konto erstellen

Jedes SolidFire Konto repräsentiert einen eindeutigen Volume-Inhaber und erhält einen eigenen Satz von Challenge-Handshake Authentication Protocol (CHAP)-Anmeldeinformationen. Sie können auf die einem Konto zugewiesenen Volumes entweder mit dem Kontonamen und den entsprechenden CHAP-Anmeldeinformationen oder über eine Volume-Zugriffsgruppe zugreifen. Einem Konto können bis zu zweitausend Volumes zugewiesen werden, aber ein Volume kann nur zu einem Konto gehören.

Erstellen Sie eine QoS-Richtlinie

Verwenden Sie SolidFire Quality of Service (QoS)-Richtlinien, wenn Sie eine standardisierte Quality of Service-Einstellung erstellen und speichern möchten, die auf viele Volumes angewendet werden kann.

Sie können QoS-Parameter pro Volume festlegen. Die Leistung jedes Volumes kann durch das Festlegen von drei konfigurierbaren Parametern sichergestellt werden, die die QoS definieren: Min IOPS, Max IOPS und Burst IOPS.

Hier sind die möglichen Minimal-, Maximal- und Burst-IOPS-Werte für die 4Kb-Blockgröße.

IOPS-Parameter	Definition	Min. Wert	Standardwert	Max. value (4 KB)
Min. IOPS	Das garantierte Leistungsniveau für ein Volume.	50	50	15000
Max. IOPS	Die Leistung wird dieses Limit nicht überschreiten.	50	15000	200.000
Burst-IOPS	Maximal zulässige IOPS in einem Kurzzeit-Burst-Szenario.	50	15000	200.000

HINWEIS Obwohl die Werte für Max IOPS und Burst IOPS auf bis zu 200.000 eingestellt werden können, ist die tatsächliche maximale Leistung eines Volumes durch die Clusternutzung und die Leistung pro Node begrenzt.

Blockgröße und Bandbreite haben einen direkten Einfluss auf die Anzahl der IOPS. Mit zunehmender Blockgröße erhöht das System die Bandbreite auf ein Niveau, das notwendig ist, um die größeren Blockgrößen zu verarbeiten. Mit zunehmender Bandbreite sinkt die Anzahl der IOPS, die das System erreichen kann. Weitere Informationen zu QoS und Leistung finden Sie unter "SolidFire Quality of Service".

SolidFire Authentifizierung

Element unterstützt zwei Methoden für die Authentifizierung: CHAP und Volume Access Groups (VAG). CHAP verwendet das CHAP-Protokoll, um den Host gegenüber dem Backend zu authentifizieren. Volume Access Groups steuern den Zugriff auf die Volumes, die sie bereitstellen. NetApp empfiehlt die Verwendung von CHAP zur Authentifizierung, da es einfacher ist und keine Skalierungsbeschränkungen hat.

HINWEIS	Trident mit dem erweiterten CSI-Provisioner unterstützt die Verwendung der CHAP-Authentifizierung. VAGs sollten nur im herkömmlichen Nicht-CSI-Betriebsmodus verwendet werden.

CHAP-Authentifizierung (Überprüfung, ob der Initiator der beabsichtigte Volume-Benutzer ist) wird nur bei kontobasierter Zugriffskontrolle 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 Initiator-Geheimnisses. Die bidirektionale CHAP-Option bietet die sicherste Methode zur Authentifizierung des Volumes, da das Volume den Host über den Kontonamen und das Initiator-Geheimnis authentifiziert und anschließend der Host das Volume über den Kontonamen und das Ziel-Geheimnis authentifiziert.

Wenn CHAP nicht aktiviert werden kann und VAGs erforderlich sind, erstellen Sie die Zugriffsgruppe und fügen Sie die Host-Initiatoren und Volumes zur Zugriffsgruppe hinzu. Jeder IQN, den Sie einer Zugriffsgruppe hinzufügen, kann auf jedes Volume in der Gruppe mit oder ohne CHAP-Authentifizierung zugreifen. Wenn der iSCSI-Initiator für die Verwendung von CHAP-Authentifizierung konfiguriert ist, wird die kontobasierte Zugriffskontrolle verwendet. Wenn der iSCSI-Initiator nicht für die Verwendung von CHAP-Authentifizierung konfiguriert ist, wird die Zugriffskontrolle über Volume Access Group verwendet.

Wo finde ich weitere Informationen?

Einige der Best-Practice-Dokumentationen sind unten aufgeführt. Suchen Sie im "NetApp Bibliothek" nach den aktuellsten Versionen.

ONTAP

- "NFS Best Practice and Implementation Guide"
- "SAN-Verwaltung" (für iSCSI)
- "iSCSI Express-Konfiguration für RHEL"

Element Software

- "Konfiguration von SolidFire für Linux"

NetApp HCI

- "NetApp HCI-Bereitstellungsvoraussetzungen"
- "Zugriff auf die NetApp Deployment Engine"

Informationen zu bewährten Anwendungspraktiken

- "Bewährte Verfahren für MySQL auf ONTAP"
- "Best Practices für MySQL auf SolidFire"
- "NetApp SolidFire und Cassandra"
- "Oracle Best Practices auf SolidFire"
- "PostgreSQL Best Practices auf SolidFire"

Nicht alle Anwendungen verfügen über spezifische Richtlinien, es ist wichtig, mit Ihrem NetApp Team zusammenzuarbeiten und die "NetApp Bibliothek" zu nutzen, um die aktuellste Dokumentation zu finden.

Trident integrieren

Um Trident zu integrieren, müssen die folgenden Design- und Architekturelemente integriert werden: Treiberauswahl und -bereitstellung, Speicherklassendesign, Design virtueller Pools, Auswirkungen von Persistent Volume Claims (PVC) auf die Speicherbereitstellung, Volumenoperationen und die Bereitstellung von OpenShift-Diensten mit Trident.

Fahrerauswahl und -bereitstellung

Wählen und implementieren Sie einen Backend-Treiber für Ihr Speichersystem.

ONTAP-Backend-Treiber

ONTAP-Backend-Treiber unterscheiden sich durch das verwendete Protokoll und die Art, wie die Volumes auf dem Speichersystem bereitgestellt werden. Daher sollten Sie sorgfältig überlegen, welchen Treiber Sie einsetzen.

Auf einer höheren Ebene gilt: Wenn Ihre Anwendung Komponenten enthält, die gemeinsam genutzten Speicher benötigen (mehrere Pods greifen auf dieselbe PVC zu), sind NAS-basierte Treiber die Standardwahl, während die blockbasierten iSCSI-Treiber den Bedarf an nicht gemeinsam genutztem Speicher abdecken. Wählen Sie das Protokoll basierend auf den Anforderungen der Anwendung und dem Erfahrungsstand der Speicher- und Infrastrukturteams. Allgemein gesprochen gibt es für die meisten Anwendungen kaum Unterschiede zwischen ihnen, sodass die Entscheidung oft davon abhängt, ob gemeinsam genutzter Speicher (bei dem mehr als ein Pod gleichzeitig Zugriff benötigt) erforderlich ist oder nicht.

Die verfügbaren ONTAP Backend-Treiber sind:

- `ontap-nas`: Jeder bereitgestellte PV ist ein vollständiger ONTAP FlexVolume.
- `ontap-nas-economy`: Jeder bereitgestellte PV ist ein Qtree, mit einer konfigurierbaren Anzahl von Qtrees pro FlexVolume (Standard ist 200).
- `ontap-nas-flexgroup`: Jeder PV wird als vollständiger ONTAP FlexGroup bereitgestellt, und alle einem SVM zugewiesenen Aggregate werden verwendet.
- `ontap-san`: Jeder bereitgestellte PV ist eine LUN innerhalb seines eigenen FlexVolume.
- `ontap-san-economy`: Jedes bereitgestellte PV ist eine LUN, mit einer konfigurierbaren Anzahl von LUNs pro FlexVolume (Standardwert ist 100).

Die Wahl zwischen den drei NAS-Treibern hat einige Auswirkungen auf die Funktionen, die der Anwendung zur Verfügung gestellt werden.

Beachten Sie, dass in den folgenden Tabellen nicht alle Funktionen über Trident verfügbar sind. Einige müssen vom Speicheradministrator nach der Bereitstellung angewendet werden, falls diese Funktionalität gewünscht ist. Die hochgestellten Fußnoten kennzeichnen die Funktionalität pro Feature und Treiber.

ONTAP NAS-Treiber	Snapshots	Klone	Dynamische Exportrichtlinien	Multi-Attach	QoS	Größe ändern	Replication
`ontap-nas`	Ja	Ja	JaFußnote:5[]	Ja	JaFußnote:1[]	Ja	JaFußnote:1[]
`ontap-nas-economy`	NO [3]	NO [3]	JaFußnote:5[]	Ja	NO [3]	Ja	NO [3]
`ontap-nas-flexgroup`	JaFußnote:1[]	NEIN	JaFußnote:5[]	Ja	JaFußnote:1[]	Ja	JaFußnote:1[]

Trident bietet 2 SAN-Treiber für ONTAP, deren Fähigkeiten unten gezeigt werden.

ONTAP SAN-Treiber	Snapshots	Klone	Multi-Attach	Bidirektionales CHAP	QoS	Größe ändern	Replication
`ontap-san`	Ja	Ja	JaFußnote:4[]	Ja	JaFußnote:1[]	Ja	JaFußnote:1[]
`ontap-san-economy`	Ja	Ja	JaFußnote:4[]	Ja	NO [3]	Ja	NO [3]

> Fußnote für die obigen Tabellen: Yes [1]: Nicht von Trident verwaltet Yes [2]: Von Trident verwaltet, aber nicht PV-granular NO [3]: Nicht von Trident verwaltet und nicht PV-granular Yes [4]: Unterstützt für Raw-Block-Volumes Yes [5]: Unterstützt von Trident

Die Merkmale, die nicht PV-granular sind, werden auf das gesamte FlexVolume angewendet, und alle PVs (d. h. qtrees oder LUNs in gemeinsamen FlexVols) teilen sich einen gemeinsamen Zeitplan.

Wie wir in den obigen Tabellen sehen können, ist vieles der Funktionalität zwischen dem `ontap-nas` und dem `ontap-nas-economy` gleich. Da jedoch der `ontap-nas-economy` Treiber die Möglichkeit einschränkt, den Zeitplan auf PV-Granularität zu steuern, kann dies insbesondere Ihre Notfallwiederherstellungs- und Backup-Planung beeinflussen. Für Entwicklungsteams, die die PVC-Klonfunktionalität auf ONTAP-Speicher nutzen möchten, ist dies nur möglich, wenn sie die `ontap-nas`, `ontap-san` oder `ontap-san-economy` Treiber verwenden.

HINWEIS	Der `solidfire-san` Treiber ist auch in der Lage, PVCs zu klonen.

Cloud Volumes ONTAP Backend-Treiber

Cloud Volumes ONTAP bietet Datenkontrolle zusammen mit Speicherfunktionen der Enterprise-Klasse für verschiedene Anwendungsfälle, einschließlich Dateifreigaben und Blockspeicher 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.

Amazon FSx for ONTAP Backend-Treiber

Amazon FSx for NetApp ONTAP ermöglicht es Ihnen, NetApp-Funktionen, Leistung und Verwaltungsfunktionen zu nutzen, mit denen Sie vertraut sind, während Sie gleichzeitig von der Einfachheit, Agilität, Sicherheit und Skalierbarkeit der Datenspeicherung auf AWS profitieren. FSx for ONTAP unterstützt viele ONTAP-Dateisystemfunktionen und Verwaltungs-APIs. Die kompatiblen Treiber für Cloud Volume ONTAP sind `ontap-nas`, `ontap-nas-economy`, `ontap-nas-flexgroup`, `ontap-san` und `ontap-san-economy`.

NetApp HCI/SolidFire Backend-Treiber

Der `solidfire-san` Treiber, der mit den NetApp HCI/SolidFire Plattformen verwendet wird, hilft dem Administrator, ein Element-Backend für Trident auf Basis von QoS-Grenzwerten zu konfigurieren. Wenn Sie Ihr Backend so gestalten möchten, dass spezifische QoS-Grenzwerte für die von Trident bereitgestellten Volumes festgelegt werden, verwenden Sie den `type` Parameter in der Backend-Datei. Der Administrator kann außerdem die Volume-Größe, die auf dem Speicher erstellt werden kann, mit dem `limitVolumeSize` Parameter beschränken. Derzeit werden Element-Speicherfunktionen wie Volume-Resize und Volume-Replikation nicht über den `solidfire-san` Treiber unterstützt. Diese Vorgänge sollten manuell über die Element Software Web-Oberfläche durchgeführt werden.

SolidFire Treiber	Snapshots	Klone	Multi-Attach	CHAP	QoS	Größe ändern	Replication
`solidfire-san`	Ja	Ja	JaFußnote:2[]	Ja	Ja	Ja	JaFußnote:1[]

> Fußnote: Yes [1]: Wird nicht von Trident verwaltet Yes [2]: Unterstützt für Raw-Block-Volumes

Azure NetApp Files-Backend-Treiber

Trident verwendet den `azure-netapp-files`-Treiber, um den "Azure NetApp Files"-Dienst zu verwalten.

Weitere Informationen zu diesem Treiber und wie Sie ihn konfigurieren können, finden Sie in "Trident Backend-Konfiguration für Azure NetApp Files".

Azure NetApp Files-Treiber	Snapshots	Klone	Multi-Attach	QoS	Erweitern	Replication
`azure-netapp-files`	Ja	Ja	Ja	Ja	Ja	JaFußnote:1[]

> Fußnote: Yes [1]: Nicht von Trident verwaltet

Speicherklassendesign

Einzelne Speicherklassen müssen konfiguriert und angewendet werden, um ein Kubernetes Storage Class-Objekt zu erstellen. In diesem Abschnitt wird erläutert, wie Sie eine Speicherklasse für Ihre Anwendung entwerfen.

Spezifische Backend-Nutzung

Innerhalb eines bestimmten Speicherklassenobjekts kann mithilfe von Filtern festgelegt werden, welcher Speicherpool oder welche Speicherpools für diese bestimmte Speicherklasse verwendet werden sollen. Drei

Filtergruppen können in der Speicherklasse festgelegt werden: `storagePools`, `additionalStoragePools`, und/oder `excludeStoragePools`.

Der `storagePools` Parameter hilft, den Speicher auf die Menge der Pools zu beschränken, die mit beliebigen angegebenen Attributen übereinstimmen. Der `additionalStoragePools` Parameter wird verwendet, um die Menge der Pools, die Trident für die Bereitstellung verwendet, zusammen mit der durch die Attribute und `storagePools` Parameter ausgewählten Menge von Pools zu erweitern. Sie können entweder einen der beiden Parameter allein oder beide zusammen verwenden, um sicherzustellen, dass die geeignete Menge an Speicherpools ausgewählt wird.

Der `excludeStoragePools`Parameter wird verwendet, um die aufgelistete Gruppe von Pools, die den Attributen entsprechen, gezielt auszuschließen.

QoS-Richtlinien emulieren

Wenn Sie Storage Classes entwerfen möchten, um Quality of Service-Richtlinien zu emulieren, erstellen Sie eine Storage Class mit dem `media` Attribut als `hdd` oder `ssd`. Basierend auf dem `media` Attribut, das in der Storage Class angegeben ist, wählt Trident das passende Backend aus, das `hdd` oder `ssd` Aggregate bereitstellt, um das Media-Attribut abzugleichen, und leitet dann die Bereitstellung der Volumes auf das spezifische Aggregat weiter. Daher können wir eine Storage Class PREMIUM erstellen, bei der das `media` Attribut als `ssd` gesetzt ist, was als PREMIUM QoS-Richtlinie klassifiziert werden kann. Wir können eine weitere Storage Class STANDARD erstellen, bei der das Media-Attribut auf `hdd` gesetzt ist, was als STANDARD QoS-Richtlinie klassifiziert werden kann. Wir könnten auch das ``IOPS'' Attribut in der Storage Class verwenden, um die Bereitstellung auf ein Element Appliance umzuleiten, das als QoS-Richtlinie definiert werden kann.

Backend basierend auf spezifischen Funktionen nutzen

Speicherklassen können so konzipiert werden, dass sie die Volume-Bereitstellung auf einem bestimmten Backend steuern, auf dem Funktionen wie Thin und Thick Provisioning, Snapshots, Klone und Verschlüsselung aktiviert sind. Um festzulegen, welcher Speicher verwendet werden soll, erstellen Sie Speicherklassen, die das entsprechende Backend mit der erforderlichen aktivierten Funktion angeben.

Virtuelle Pools

Virtuelle Pools sind für alle Trident Backends verfügbar. Sie können virtuelle Pools für jedes Backend definieren, wobei Sie jeden von Trident bereitgestellten Treiber verwenden können.

Virtuelle Pools ermöglichen es einem Administrator, eine Abstraktionsebene über Backends zu erstellen, die über Storage Classes referenziert werden können, um mehr Flexibilität und eine effiziente Platzierung von Volumes auf den Backends zu erreichen. Verschiedene Backends können mit derselben Serviceklasse definiert werden. Außerdem können mehrere Speicherpools auf demselben Backend, aber mit unterschiedlichen Eigenschaften erstellt werden. Wenn eine Storage Class mit einem Selektor mit den spezifischen Labels konfiguriert wird, wählt Trident ein Backend aus, das allen Selektor-Labels entspricht, um das Volume zu platzieren. Wenn die Selektor-Labels der Storage Class mit mehreren Speicherpools übereinstimmen, wählt Trident einen davon aus, um das Volume bereitzustellen.

Design des virtuellen Pools

Beim Erstellen eines Backends können Sie üblicherweise eine Reihe von Parametern festlegen. Es war dem Administrator nicht möglich, ein weiteres Backend mit denselben Speicherzugangsdaten und einem anderen Parametersatz zu erstellen. Mit der Einführung virtueller Pools wurde dieses Problem behoben. Ein virtueller Pool ist eine Abstraktionsebene, die zwischen dem Backend und der Kubernetes Storage Class eingeführt wurde, sodass der Administrator Parameter zusammen mit Labels definieren kann, die über Kubernetes

Storage Classes als Selektor backendunabhängig referenziert werden können. Virtuelle Pools können für alle unterstützten NetApp Backends mit Trident definiert werden. Diese Liste umfasst SolidFire/NetApp HCI, ONTAP sowie Azure NetApp Files.

HINWEIS	Bei der Definition virtueller Pools wird empfohlen, die Reihenfolge bestehender virtueller Pools in einer Backend-Definition nicht zu ändern. Es ist außerdem ratsam, Attribute eines bestehenden virtuellen Pools nicht zu bearbeiten oder zu ändern und stattdessen einen neuen virtuellen Pool zu definieren.

Emulieren verschiedener Servicelevel/QoS

Es ist möglich, virtuelle Pools zur Emulation von Dienstklassen zu entwerfen. Anhand der Implementierung virtueller Pools für Cloud Volume Service für Azure NetApp Files untersuchen wir, wie wir verschiedene Dienstklassen einrichten können. Konfigurieren Sie das Azure NetApp Files-Backend mit mehreren Labels, die unterschiedliche Leistungsstufen repräsentieren. Setzen Sie den `servicelevel` Aspekt auf die entsprechende Leistungsstufe und fügen Sie unter jedem Label weitere erforderliche Aspekte hinzu. Erstellen Sie nun verschiedene Kubernetes Storage Classes, die jeweils unterschiedlichen virtuellen Pools zugeordnet werden. Mit dem `parameters.selector` Feld gibt jede StorageClass an, welche virtuellen Pools zum Hosten eines Volumes verwendet werden können.

Zuweisung eines bestimmten Satzes von Aspekten

Aus einem einzigen Storage-Backend lassen sich mehrere virtuelle Pools mit jeweils spezifischen Aspekten erstellen. Konfigurieren Sie dazu das Backend mit mehreren Labels und legen Sie unter jedem Label die benötigten Aspekte fest. Erstellen Sie anschließend verschiedene Kubernetes Storage Classes, indem Sie das `parameters.selector` Feld verwenden, das den verschiedenen virtuellen Pools zugeordnet wird. Die auf dem Backend bereitgestellten Volumes enthalten dann die im jeweiligen virtuellen Pool definierten Aspekte.

PVC-Eigenschaften, die die Speicherbereitstellung beeinflussen

Einige Parameter außerhalb der angeforderten Speicherklasse können den Trident-Bereitstellungsentscheidungsprozess bei der Erstellung eines PVC beeinflussen.

Zugriffsmodus

Bei der Anforderung von Speicherplatz über eine PVC ist der Zugriffsmodus eines der Pflichtfelder. Der gewünschte Modus kann Einfluss darauf haben, welches Backend für die Verarbeitung der Speicheranforderung ausgewählt wird.

Trident versucht, das verwendete Speicherprotokoll der angegebenen Zugriffsmethode gemäß der folgenden Matrix zuzuordnen. Dies ist unabhängig von der zugrunde liegenden Speicherplattform.

	ReadWriteOnce	ReadOnlyMany	ReadWriteMany
iSCSI	Ja	Ja	Ja (Rohblock)
NFS	Ja	Ja	Ja

Eine Anfrage für eine ReadWriteMany PVC, die an eine Trident-Bereitstellung ohne konfiguriertes NFS-Backend gesendet wird, führt dazu, dass kein Volume bereitgestellt wird. Aus diesem Grund sollte der Anfragende den Zugriffsmodus verwenden, der für seine Anwendung geeignet ist.

Volumenoperationen

Persistente Volumes ändern

Persistente Volumes sind in Kubernetes – mit zwei Ausnahmen – unveränderliche Objekte. Nach ihrer Erstellung können die Reclaim-Policy und die Größe angepasst werden. Dies verhindert jedoch nicht, dass bestimmte Aspekte des Volumes außerhalb von Kubernetes geändert werden. Dies kann wünschenswert sein, um das Volume für spezifische Anwendungen anzupassen, um sicherzustellen, dass die Kapazität nicht versehentlich verbraucht wird, oder einfach, um das Volume aus beliebigen Gründen auf einen anderen Speichercontroller zu verschieben.

HINWEIS	Kubernetes in-tree Provisioner unterstützen derzeit keine Größenänderung von NFS-, iSCSI- oder FC-PVs. Trident unterstützt die Erweiterung von NFS-, iSCSI- und FC-Volumes.

Die Verbindungsdetails des PV können nach der Erstellung nicht mehr geändert werden.

Erstellen Sie bedarfsgesteuerte Volume-Snapshots

Trident unterstützt die bedarfsgesteuerte Erstellung von Volume-Snapshots und die Erstellung von PVCs aus Snapshots mithilfe des CSI-Frameworks. Snapshots bieten eine komfortable Methode zur Verwaltung zeitpunktgenauer Kopien der Daten und haben einen vom Quell-PV in Kubernetes unabhängigen Lebenszyklus. Diese Snapshots können zum Klonen von PVCs verwendet werden.

Volumes aus Snapshots erstellen

Trident unterstützt auch die Erstellung von PersistentVolumes aus Volume-Snapshots. Dazu erstellen Sie einfach eine PersistentVolumeClaim und geben den `datasource` gewünschten Snapshot an, aus dem das Volume erstellt werden soll. Trident wird diese PVC verarbeiten, indem ein Volume mit den auf dem Snapshot vorhandenen Daten erstellt wird. Mit dieser Funktion ist es möglich, Daten regionsübergreifend zu duplizieren, Testumgebungen zu erstellen, ein beschädigtes oder beschädigtes Produktionsvolume vollständig zu ersetzen oder bestimmte Dateien und Verzeichnisse abzurufen und auf ein anderes angeschlossenes Volume zu übertragen.

Verschieben Sie Volumes im Cluster

Speicheradministratoren haben die Möglichkeit, Volumes zwischen Aggregaten und Controllern im ONTAP Cluster unterbrechungsfrei für den Speicherkonsumenten zu verschieben. Dieser Vorgang hat keine Auswirkungen auf Trident oder den Kubernetes-Cluster, solange das Zielaggregat eines ist, auf das die von Trident verwendete SVM Zugriff hat. Wichtig ist, dass, wenn das Aggregat neu zur SVM hinzugefügt wurde, das Backend durch erneutes Hinzufügen zu Trident aktualisiert werden muss. Dadurch wird Trident veranlasst, die SVM neu zu inventarisieren, sodass das neue Aggregat erkannt wird.

Das Verschieben von Volumes zwischen Backends wird von Trident nicht automatisch unterstützt. Dies gilt für Verschiebungen zwischen SVMs im selben Cluster, zwischen Clustern oder auf eine andere Speicherplattform (selbst wenn dieses Speichersystem mit Trident verbunden ist).

Wenn ein Volume an einen anderen Speicherort kopiert wird, kann die Volume-Importfunktion verwendet werden, um die aktuellen Volumes in Trident zu importieren.

Volumen erweitern

Trident unterstützt die Größenänderung von NFS-, iSCSI- und FC-PVs. Dadurch können Benutzer ihre Volumes direkt über die Kubernetes-Schicht anpassen. Die Volume-Erweiterung ist für alle gängigen NetApp

Speicherplattformen möglich, einschließlich ONTAP und SolidFire/NetApp HCI-Backends. Um eine spätere Erweiterung zu ermöglichen, setzen Sie `allowVolumeExpansion` auf `true` in Ihrer StorageClass, die dem Volume zugeordnet ist. Wann immer das Persistent Volume vergrößert werden muss, bearbeiten Sie die `spec.resources.requests.storage` Annotation im Persistent Volume Claim auf die gewünschte Volume-Größe. Trident kümmert sich dann automatisch um die Größenänderung des Volumes im Speichercluster.

Ein vorhandenes Volume in Kubernetes importieren

Der Volume-Import ermöglicht das Importieren eines vorhandenen Speichervolumes in eine Kubernetes-Umgebung. Dies wird derzeit von den `ontap-nas`, `ontap-nas-flexgroup`, `solidfire-san` und `azure-netapp-files` Treibern unterstützt. Diese Funktion ist nützlich beim Portieren einer bestehenden Anwendung nach Kubernetes oder während Notfallwiederherstellungsszenarien.

Bei Verwendung der 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, damit es von Trident verwaltet wird. Die im Import-Volume-Befehl verwendete PVC-YAML- oder JSON-Datei verweist auf eine Storage-Class, die Trident als Provisioner identifiziert. Bei Verwendung eines NetApp HCI/SolidFire Backends stellen Sie sicher, dass die Volume-Namen eindeutig sind. Wenn die Volume-Namen doppelt vorhanden sind, klonen Sie das Volume unter einem eindeutigen Namen, damit die Volume-Importfunktion zwischen ihnen unterscheiden kann.

Wenn der `azure-netapp-files` Treiber verwendet wird, verwenden Sie den Befehl `tridentctl import volume <backend-name> <volume path> -f /path/pvc.yaml`, um das Volume in Kubernetes zu importieren, damit es von Trident verwaltet wird. Dadurch wird eine eindeutige Volume-Referenz sichergestellt.

Wenn der oben stehende Befehl ausgeführt wird, findet Trident das Volume im Backend und liest dessen Größe aus. Es wird automatisch die konfigurierte Größe des PVC-Volumes hinzugefügt (und bei Bedarf überschrieben). Anschließend erstellt Trident das neue PV und Kubernetes bindet das PVC an das PV.

Wenn ein Container so bereitgestellt wurde, dass er die spezifische importierte PVC benötigt, bleibt er im Status „Ausstehend“, bis das PVC/PV-Paar über den Volume-Importprozess gebunden ist. Nachdem das PVC/PV-Paar gebunden ist, sollte der Container starten, sofern keine weiteren Probleme auftreten.

Registrierungsdienst

Die Bereitstellung und Verwaltung des Speichers für die Registry wurde auf "netapp.io" in der "Blog" dokumentiert.

Protokollierungsdienst

Wie andere OpenShift-Dienste wird der Protokollierungsdienst mithilfe von Ansible mit den in der Inventardatei, auch Hosts genannt, bereitgestellten Konfigurationsparametern bereitgestellt, die dem Playbook übergeben wird. Es gibt zwei Installationsmethoden, die behandelt werden: die Bereitstellung der Protokollierung während der anfänglichen OpenShift-Installation und die Bereitstellung der Protokollierung, nachdem OpenShift installiert wurde.

ACHTUNG	Ab Red Hat OpenShift Version 3.9 empfiehlt die offizielle Dokumentation NFS für den Protokollierungsdienst nicht zu verwenden, aufgrund von Bedenken hinsichtlich Datenbeschädigung. Dies basiert auf Red Hat Tests ihrer Produkte. Der ONTAP NFS-Server hat diese Probleme nicht und kann problemlos eine Protokollierungsbereitstellung unterstützen. Letztendlich liegt die Wahl des Protokolls für den Protokollierungsdienst bei Ihnen, aber beide funktionieren hervorragend mit NetApp Plattformen und es gibt keinen Grund, NFS zu vermeiden, wenn dies Ihre Präferenz ist.

Wenn Sie NFS mit dem Protokollierungsdienst verwenden, müssen Sie die Ansible-Variable `openshift_enable_unsupported_configurations` auf `true` setzen, um zu verhindern, dass die Installation fehlschlägt.

Los geht's

Der Protokollierungsdienst kann optional sowohl für Anwendungen als auch für die Kernoperationen des OpenShift-Clusters bereitgestellt werden. Wenn Sie sich entscheiden, die Protokollierung für den Betrieb bereitzustellen, indem Sie die Variable `openshift_logging_use_ops` als `true` angeben, werden zwei Instanzen des Dienstes erstellt. Die Variablen, die die Protokollierungsinstanz für den Betrieb steuern, enthalten „ops“, während die Instanz für Anwendungen dies nicht tut.

Die Konfiguration der Ansible-Variablen entsprechend der Bereitstellungsmethode ist wichtig, um sicherzustellen, dass der korrekte Speicher von den zugrunde liegenden Diensten genutzt wird. Sehen wir uns die Optionen für jede der Bereitstellungsmethoden an.

HINWEIS Die folgenden Tabellen enthalten ausschließlich die für die Speicherkonfiguration im Zusammenhang mit dem Protokollierungsdienst relevanten Variablen. Weitere Optionen finden Sie in "Red Hat OpenShift Protokollierungsdokumentation", die Sie prüfen, konfigurieren und entsprechend Ihrer Bereitstellung verwenden sollten.

Die Variablen in der folgenden Tabelle führen dazu, dass das Ansible-Playbook ein PV und ein PVC für den Protokollierungsdienst anhand der angegebenen Details erstellt. Diese Methode ist deutlich weniger flexibel als die Verwendung des Playbooks zur Komponenteninstallation nach der OpenShift-Installation, jedoch ist sie eine Option, wenn bereits Volumes verfügbar sind.

Variable	Details
`openshift_logging_storage_kind`	Setzen Sie `nfs` , damit das Installationsprogramm ein NFS-PV für den Protokollierungsdienst erstellt.
`openshift_logging_storage_host`	Der Hostname oder die IP-Adresse des NFS-Hosts. Dies sollte auf die dataLIF für Ihre virtuelle Maschine eingestellt werden.
`openshift_logging_storage_nfs_directory`	Der Mount-Pfad für den NFS-Export. Wenn das Volume beispielsweise als `/openshift_logging` eingebunden ist, verwenden Sie diesen Pfad für diese Variable.
`openshift_logging_storage_volume_name`	Der Name, z. B. `pv_ose_logs`, des zu erstellenden PV.
`openshift_logging_storage_volume_size`	Die Größe des NFS-Exports, zum Beispiel `100Gi`.

Wenn Ihr OpenShift-Cluster bereits läuft und somit Trident bereitgestellt und konfiguriert wurde, kann das Installationsprogramm die Volumes mithilfe der dynamischen Bereitstellung erstellen. Die folgenden Variablen müssen konfiguriert werden.

Variable	Details
`openshift_logging_es_pvc_dynamic`	Auf „true“ setzen, um dynamisch bereitgestellte Volumes zu verwenden.
`openshift_logging_es_pvc_storage_class_name`	Der Name der Speicherklasse, die im PVC verwendet wird.

Variable	Details
`openshift_logging_es_pvc_size`	Die im PVC angeforderte Größe des Volumens.
`openshift_logging_es_pvc_prefix`	Ein Präfix für die von dem Protokollierungsdienst verwendeten PVCs.
`openshift_logging_es_ops_pvc_dynamic`	Auf `true` setzen, um dynamisch bereitgestellte Volumes für die Ops-Protokollierungsinstanz zu verwenden.
`openshift_logging_es_ops_pvc_storage_class_name`	Der Name der Speicherklasse für die Protokollierungsinstanz.
`openshift_logging_es_ops_pvc_size`	Die Größe der Volumenanforderung für die ops-Instanz.
`openshift_logging_es_ops_pvc_prefix`	Ein Präfix für die PVCs der ops-Instanz.

Stellen Sie den Protokollierungs-Stack bereit

Wenn Sie die Protokollierung als Teil des anfänglichen OpenShift-Installationsprozesses bereitstellen, müssen Sie lediglich dem Standard-Bereitstellungsprozess folgen. Ansible konfiguriert und stellt die benötigten Dienste und OpenShift-Objekte bereit, sodass der Dienst verfügbar ist, sobald Ansible abgeschlossen ist.

Wenn Sie jedoch nach der Erstinstallation eine Bereitstellung durchführen, muss das Komponenten-Playbook von Ansible verwendet werden. Dieser Prozess kann sich je nach Version von OpenShift geringfügig ändern, daher sollten Sie unbedingt "Red Hat OpenShift Container Platform 3.11 Dokumentation" für Ihre Version lesen und befolgen.

Metrikdienst

Der Metrikdienst liefert dem Administrator wertvolle Informationen über Status, Ressourcenauslastung und Verfügbarkeit des OpenShift Clusters. Er ist außerdem für die automatische Pod-Skalierung erforderlich und viele Organisationen nutzen die Daten des Metrikdienstes für ihre Kostenverrechnungs- und/oder Kostenanzeigeanwendungen.

Wie beim Protokollierungsdienst und bei OpenShift insgesamt wird Ansible verwendet, um den Metrikdienst bereitzustellen. Ebenso wie der Protokollierungsdienst kann der Metrikdienst entweder während der Ersteinrichtung des Clusters oder nach dessen Inbetriebnahme mithilfe der Komponenteninstallationsmethode bereitgestellt werden. Die folgenden Tabellen enthalten die Variablen, die bei der Konfiguration des persistenten Speichers für den Metrikdienst wichtig sind.

HINWEIS	Die folgenden Tabellen enthalten nur die Variablen, die für die Speicherkonfiguration im Zusammenhang mit dem Metrics Service relevant sind. In der Dokumentation finden Sie viele weitere Optionen, die geprüft, konfiguriert und entsprechend Ihrer Implementierung verwendet werden sollten.

Variable	Details
`openshift_metrics_storage_kind`	Setzen Sie `nfs` , damit das Installationsprogramm ein NFS-PV für den Protokollierungsdienst erstellt.
`openshift_metrics_storage_host`	Der Hostname oder die IP-Adresse des NFS-Hosts. Dieser Wert sollte auf die dataLIF für Ihre SVM eingestellt werden.

Variable	Details
`openshift_metrics_storage_nfs_directory`	Der Mount-Pfad für den NFS-Export. Wenn das Volume beispielsweise als `/openshift_metrics` eingebunden ist, verwenden Sie diesen Pfad für diese Variable.
`openshift_metrics_storage_volume_name`	Der Name, z. B. `pv_ose_metrics`, des zu erstellenden PV.
`openshift_metrics_storage_volume_size`	Die Größe des NFS-Exports, zum Beispiel `100Gi`.

Wenn Ihr OpenShift-Cluster bereits läuft und somit Trident bereitgestellt und konfiguriert wurde, kann das Installationsprogramm die Volumes mithilfe der dynamischen Bereitstellung erstellen. Die folgenden Variablen müssen konfiguriert werden.

Variable	Details
`openshift_metrics_cassandra_pvc_prefix`	Ein Präfix, das für die Metrik-PVCs verwendet werden soll.
`openshift_metrics_cassandra_pvc_size`	Die Größe der anzufordernden Volumes.
`openshift_metrics_cassandra_storage_type`	Der Typ des Speichers, der für Metriken verwendet werden soll, muss auf dynamisch gesetzt werden, damit Ansible PVCs mit der entsprechenden Speicherklasse erstellt.
`openshift_metrics_cassanda_pvc_storage_class_name`	Der Name der zu verwendenden Speicherklasse.

Stellen Sie den Metrikdienst bereit

Mit den entsprechenden Ansible-Variablen, die in Ihrer Hosts/Inventory-Datei definiert sind, stellen Sie den Dienst mit Ansible bereit. Wenn Sie während der OpenShift-Installation bereitstellen, wird das PV automatisch erstellt und verwendet. Wenn Sie mit den Komponenten-Playbooks nach der OpenShift-Installation bereitstellen, erstellt Ansible alle benötigten PVCs und nachdem Trident Speicher dafür bereitgestellt hat, wird der Dienst bereitgestellt.

Die oben genannten Variablen und der Bereitstellungsprozess können sich mit jeder Version von OpenShift ändern. Stellen Sie sicher, dass Sie "Red Hat's OpenShift Implementierungs-Leitfaden" für Ihre Version überprüfen und befolgen, damit diese für Ihre Umgebung konfiguriert ist.

Datenschutz und Notfallwiederherstellung

Informieren Sie sich über die Schutz- und Wiederherstellungsoptionen für Trident und Volumes, die mit Trident erstellt wurden. Sie sollten für jede Anwendung mit Persistenzanforderung eine Strategie zur Datensicherung und Wiederherstellung haben.

Trident-Replikation und -Wiederherstellung

Sie können eine Sicherungskopie erstellen, um Trident im Katastrophenfall wiederherzustellen.

Trident Replikation

Trident verwendet Kubernetes CRDs, um seinen eigenen Zustand zu speichern und zu verwalten, und das Kubernetes Cluster etcd, um seine Metadaten zu speichern.

Schritte

1. Sichern Sie das etcd des Kubernetes-Clusters mit "Kubernetes: Sichern eines etcd-Clusters".
2. Platzieren Sie die Sicherungsdateien auf einem FlexVol volume

HINWEIS	NetApp empfiehlt, die SVM, in der sich die FlexVol befindet, mit einer SnapMirror Beziehung zu einer anderen SVM zu schützen.

Trident Wiederherstellung

Mithilfe von Kubernetes-CRDs und dem etcd-Snapshot des Kubernetes-Clusters können Sie Trident wiederherstellen.

Schritte

1. Vom Ziel-SVM aus binden Sie das Volume, das die Kubernetes etcd-Datendateien und Zertifikate enthält, auf dem Host ein, der als Master-Knoten eingerichtet werden soll.
2. Kopieren Sie alle erforderlichen Zertifikate, die sich auf den Kubernetes-Cluster beziehen, unter `/etc/kubernetes/pki` und die etcd-Mitgliedsdateien unter `/var/lib/etcd`.
3. Stellen Sie den Kubernetes-Cluster aus dem etcd-Backup mithilfe von "Kubernetes: Wiederherstellen eines etcd-Clusters" wieder her.
4. Führen Sie `kubectl get crd` aus, um zu überprüfen, ob alle Trident benutzerdefinierten Ressourcen geladen wurden, und rufen Sie die Trident Objekte ab, um zu prüfen, ob alle Daten verfügbar sind.

SVM-Replikation und -Wiederherstellung

Trident kann keine Replikationsbeziehungen konfigurieren, jedoch kann der Speicheradministrator "ONTAP SnapMirror" verwenden, um eine SVM zu replizieren.

Im Katastrophenfall können Sie die SnapMirror Ziel-SVM aktivieren, um die Datenbereitstellung aufzunehmen. Sie können wieder auf die primäre SVM umschalten, wenn die Systeme wiederhergestellt sind.

Über diese Aufgabe

Beachten Sie Folgendes bei der Verwendung der SnapMirror SVM-Replikationsfunktion:

- Sie sollten für jede SVM mit aktiviertem SVM-DR ein eigenes Backend erstellen.
- Konfigurieren Sie die Speicherklassen so, dass die replizierten Backends nur bei Bedarf ausgewählt werden, um zu vermeiden, dass Volumes, die keine Replikation benötigen, auf den Backends bereitgestellt werden, die SVM-DR unterstützen.
- Anwendungsadministratoren sollten die zusätzlichen Kosten und die Komplexität verstehen, die mit der Replikation verbunden sind, und ihren Wiederherstellungsplan sorgfältig prüfen, bevor sie mit diesem Prozess beginnen.

SVM-Replikation

Sie können "ONTAP: SnapMirror SVM-Replikation" verwenden, um die SVM-Replikationsbeziehung zu erstellen.

SnapMirror ermöglicht es Ihnen, Optionen festzulegen, um zu steuern, was repliziert werden soll. Sie müssen wissen, welche Optionen Sie ausgewählt haben, wenn Sie SVM-Wiederherstellung mit Trident durchführen.

- "-identity-preserve true" repliziert die gesamte SVM-Konfiguration.
- "-discard-configs Netzwerk" schließt LIFs und zugehörige Netzwerkeinstellungen aus.
- "-identity-preserve false" Repliziert werden nur die Volumes und die Sicherheitskonfiguration.

SVM-Wiederherstellung mit Trident

Trident erkennt SVM-Ausfälle nicht automatisch. Im Katastrophenfall kann der Administrator das Trident Failover auf die neue SVM manuell einleiten.

Schritte

1. Brechen Sie geplante und laufende SnapMirror Übertragungen ab, trennen Sie die Replikationsbeziehung, stoppen Sie die Quell-SVM und aktivieren Sie anschließend die SnapMirror Ziel-SVM.
2. Wenn Sie `-identity-preserve false` oder `-discard-config network` bei der Konfiguration Ihrer SVM-Replikation angegeben haben, aktualisieren Sie die `managementLIF` und `dataLIF` in der Trident-Backend-Definitionsdatei.
3. Bestätigen `storagePrefix` ist in der Trident-Backend-Definitionsdatei vorhanden. Dieser Parameter kann nicht geändert werden. Das Weglassen von `storagePrefix` führt dazu, dass das Backend-Update fehlschlägt.
4. Aktualisieren Sie alle erforderlichen Backends, um den neuen Ziel-SVM-Namen widerzuspiegeln, indem Sie Folgendes verwenden:

   ```
   ./tridentctl update backend <backend-name> -f <backend-json-file> -n
   <namespace>
   ```

5. Wenn Sie `-identity-preserve false` oder `discard-config network` angegeben haben, müssen Sie alle Anwendungspods neu starten.

HINWEIS	Wenn Sie `-identity-preserve true` angegeben haben, beginnen alle von Trident bereitgestellten Volumes mit der Datenbereitstellung, sobald die Ziel-SVM aktiviert wird.

Volumenreplikation und Wiederherstellung

Trident kann keine SnapMirror Replikationsbeziehungen konfigurieren, jedoch kann der Speicheradministrator "ONTAP SnapMirror Replikation und Wiederherstellung" verwenden, um von Trident erstellte Volumes zu replizieren.

Anschließend können Sie die wiederhergestellten Volumes mit Trident importieren "tridentctl volume import".

HINWEIS	Import wird auf `ontap-nas-economy`, `ontap-san-economy` oder `ontap-flexgroup-economy` Treibern nicht unterstützt.

Snapshot-Datenschutz

Sie können Daten mit folgenden Methoden schützen und wiederherstellen:

- Ein externer Snapshot-Controller und CRDs zum Erstellen von Kubernetes-Volume-Snapshots von Persistent Volumes (PVs).

  "Volume Snapshots"

- ONTAP Snapshots, um den gesamten Inhalt eines Volumes wiederherzustellen oder einzelne Dateien oder LUNs wiederherzustellen.

  "ONTAP Snapshots"

Automatisierung des Failovers von zustandsbehafteten Anwendungen mit Trident

Die Force-Detach-Funktion von Trident ermöglicht es Ihnen, Volumes automatisch von fehlerhaften Knoten in einem Kubernetes-Cluster zu trennen, wodurch Datenbeschädigung verhindert und die Anwendungsverfügbarkeit sichergestellt wird. Diese Funktion ist besonders nützlich in Szenarien, in denen Knoten nicht mehr reagieren oder für Wartungsarbeiten offline genommen werden.

Details zum erzwungenen Abtrennen

Die erzwungene Trennung ist für `ontap-san`, `ontap-san-economy`, `ontap-nas` und `ontap-nas-economy` verfügbar. Bevor Sie die erzwungene Trennung aktivieren, muss das nicht-graceful node shutdown (NGNS) im Kubernetes-Cluster aktiviert sein. NGNS ist standardmäßig für Kubernetes 1.28 und höher aktiviert. Weitere Informationen finden sich unter "Kubernetes: Nicht ordnungsgemäßes Herunterfahren eines Knotens".

HINWEIS	Bei Verwendung des `ontap-nas` oder `ontap-nas-economy`-Treibers müssen Sie den `autoExportPolicy`-Parameter in der Backend-Konfiguration auf `true` setzen, damit Trident den Zugriff vom Kubernetes-Knoten mit dem angewendeten Taint mithilfe verwalteter Exportrichtlinien einschränken kann.
WARNUNG	Da Trident auf Kubernetes NGNS basiert, sollten Sie `out-of-service`Taints von einem fehlerhaften Knoten erst dann entfernen, wenn alle nicht tolerierbaren Workloads neu geplant wurden. Das unbedachte Anwenden oder Entfernen von Taints kann den Schutz der Backend-Daten gefährden.

Wenn der Kubernetes-Clusteradministrator den `node.kubernetes.io/out-of-service=nodeshutdown:NoExecute` Taint auf den Knoten angewendet hat und `enableForceDetach` auf `true` gesetzt ist, ermittelt Trident den Knotenstatus und:

1. Stoppen Sie den Backend-I/O-Zugriff für Volumes, die an diesem Node gemountet sind.
2. Markieren Sie das Trident node object als `dirty` (nicht sicher für neue Veröffentlichungen).

HINWEIS	Der Trident-Controller lehnt neue Veröffentlichungsanforderungen für Volumes ab, bis der Knoten als `dirty`vom Trident-Knotenpod erneut qualifiziert wurde. Alle Workloads, die mit einem eingebundenen PVC geplant sind (auch nachdem der Clusterknoten fehlerfrei und bereit ist), werden nicht akzeptiert, bis Trident den Knoten `clean` (sicher für neue Veröffentlichungen) verifizieren kann.

Wenn die Knotenintegrität wiederhergestellt ist und die Taint entfernt wurde, wird Trident:

1. Identifizieren und bereinigen Sie veraltete veröffentlichte Pfade auf dem Node.
2. Wenn sich der Knoten in einem `cleanable` Zustand befindet (die Außerbetriebnahme-Warnung wurde entfernt und der Knoten befindet sich im `Ready` Zustand) und alle veralteten, veröffentlichten Pfade sauber sind, wird Trident den Knoten als `clean` wieder zulassen und neue veröffentlichte Volumes auf dem Knoten erlauben.

Details zum automatischen Failover

Sie können den Prozess der erzwungenen Trennung durch die Integration mit "Node Health Check (NHC) Operator" automatisieren. Wenn ein Knotenausfall auftritt, löst NHC die Trident-Knotenbehebung (TNR) und die erzwungene Trennung automatisch aus, indem ein TridentNodeRemediation CR im Trident-Namensraum erstellt wird, der den ausgefallenen Knoten definiert. TNR wird nur bei einem Knotenausfall erstellt und von NHC entfernt, sobald der Knoten wieder online ist oder gelöscht wurde.

Fehlgeschlagener Node-Pod-Entfernungsprozess

Automated-failover wählt die Workloads aus, die vom ausgefallenen Knoten entfernt werden sollen. Wenn ein TNR erstellt wird, markiert der TNR-Controller den Knoten als dirty, verhindert die Veröffentlichung neuer Volumes und beginnt mit dem Entfernen von Force-Detach-unterstützten Pods und deren Volume-Anhängen.

Alle von Force-Detach unterstützten Volumes/PVCs werden auch von Automated-Failover unterstützt:

- NAS- und NAS-economy-Volumes unter Verwendung von Auto-Export-Richtlinien (SMB wird noch nicht unterstützt).
- SAN- und SAN-economy-Volumes.

Siehe Details zum erzwungenen Abtrennen.

Standardverhalten:

- Pods, die von force-detach unterstützte Volumes verwenden, werden vom ausgefallenen Knoten entfernt. Kubernetes wird diese auf einem fehlerfreien Knoten neu einplanen.
- Pods, die ein von force-detach nicht unterstütztes Volume verwenden, einschließlich nicht-Trident-Volumes, werden nicht vom ausgefallenen Knoten entfernt.
- Stateless Pods (nicht PVCs) werden nicht vom ausgefallenen Knoten entfernt, es sei denn, die Pod-Annotation `trident.netapp.io/podRemediationPolicy: delete` gesetzt ist.

Überschreiben des Pod-Entfernungsverhaltens:

Das Verhalten bei der Pod-Entfernung kann mithilfe einer Pod-Annotation angepasst werden: `trident.netapp.io/podRemediationPolicy[retain, delete]`. Diese Annotationen werden geprüft und verwendet, wenn ein Failover auftritt. Wenden Sie Annotationen auf die Kubernetes

Deployment/ReplicaSet Pod-Spezifikation an, um zu verhindern, dass die Annotation nach einem Failover verschwindet:

- `retain` - Der Pod wird während eines automatischen Failovers NICHT vom ausgefallenen Knoten entfernt.
- `delete` - Der Pod wird bei einem automatischen Failover vom ausgefallenen Knoten entfernt.

Diese Annotationen können auf jeden Pod angewendet werden.

WARNUNG	• E/A-Operationen werden nur auf ausgefallenen Knoten für Volumes blockiert, die force-detach unterstützen. • Für Volumes, die das erzwungene Trennen nicht unterstützen, besteht das Risiko von Datenbeschädigung und Multi-Attach-Problemen.

TridentNodeRemediation CR

Der TridentNodeRemediation (TNR) CR definiert einen ausgefallenen Knoten. Der Name des TNR ist der Name des ausgefallenen Knotens.

Beispiel TNR:

```
apiVersion: trident.netapp.io/v1
kind: TridentNodeRemediation
metadata:
  name: <K8s-node-name>
spec: {}
```

TNR states: Verwenden Sie die folgenden Befehle, um den Status der TNRs anzuzeigen:
`kubectl get tnr <name> -n <trident-namespace>`

TNRs können sich in einem der folgenden Zustände befinden:

- *Behebung*:
  - Den Backend-E/A-Zugriff für Volumes, die von force-detach unterstützt und an diesen Knoten angehängt wurden, einstellen.
  - Das Trident-Knotenobjekt ist als „dirty“ markiert (nicht sicher für neue Veröffentlichungen).
  - Entfernen Sie Pods und Volume-Anhänge vom Knoten
- *NodeRecoveryPending*:
  - Der Controller wartet darauf, dass der Knoten wieder online geht.
  - Sobald der Knoten online ist, stellt publish-enforcement sicher, dass der Knoten sauber und bereit für neue Volume-Veröffentlichungen ist.
- Wenn der Knoten aus K8s gelöscht wird, entfernt der TNR-Controller den TNR und stellt die Abstimmung ein.
- *Erfolgreich*:
  - Alle Sanierungs- und Wiederherstellungsmaßnahmen wurden erfolgreich abgeschlossen. Der Knoten ist bereinigt und bereit für neue Volume-Veröffentlichungen.

- *Fehlgeschlagen*:
  - Nicht behebbarer Fehler. Fehlergründe sind im Feld status.message des CR festgelegt.

Automatisches Failover aktivieren

Voraussetzungen:

- Stellen Sie sicher, dass die erzwungene Trennung aktiviert ist, bevor Sie das automatische Failover aktivieren. Weitere Informationen finden sich unter Details zum erzwungenen Abtrennen.
- Installieren Sie Node Health Check (NHC) im Kubernetes-Cluster.
  - "Installiere operator-sdk".
  - Installieren Sie Operator Lifecycle Manager (OLM) im Cluster, falls noch nicht installiert: `operator-sdk olm install`.
  - Installieren Sie den Node Health check Operator: `kubectl create -f https://operatorhub.io/install/node-healthcheck-operator.yaml`.

HINWEIS	Sie können auch alternative Methoden zur Erkennung von Knotenausfällen verwenden, wie im Integration kundenspezifischer Lösungen zur Überprüfung des Knotenzustands Abschnitt unten angegeben.

Weitere Informationen finden Sie unter "Node Health Check Operator".

Schritte

1. Erstellen Sie einen NodeHealthCheck (NHC) CR im Trident-Namespace, um die Worker-Knoten im Cluster zu überwachen. Beispiel:

```
apiVersion: remediation.medik8s.io/v1alpha1
kind: NodeHealthCheck
metadata:
  name: <CR name>
spec:
  selector:
    matchExpressions:
      - key: node-role.kubernetes.io/control-plane
        operator: DoesNotExist
      - key: node-role.kubernetes.io/master
        operator: DoesNotExist
  remediationTemplate:
    apiVersion: trident.netapp.io/v1
    kind: TridentNodeRemediationTemplate
    namespace: <Trident installation namespace>
    name: trident-node-remediation-template
  minHealthy: 0 # Trigger force-detach upon one or more node failures
  unhealthyConditions:
    - type: Ready
      status: "False"
      duration: 0s
    - type: Ready
      status: Unknown
      duration: 0s
```

2. Wenden Sie die Knoten-Integritätsprüfung CR im `trident` Namespace an.

```
kubectl apply -f <nhc-cr-file>.yaml -n <trident-namespace>
```

Der oben genannte CR ist so konfiguriert, dass er die K8s-Worker-Knoten auf die Knotenzustände Ready: false und Unknown überwacht. Automated-Failover wird ausgelöst, sobald ein Knoten in den Zustand Ready: false oder Ready: Unknown wechselt.

The `unhealthyConditions` in the CR verwendet eine Karenzzeit von 0 Sekunden. Dies führt dazu, dass das automatisierte Failover sofort ausgelöst wird, sobald K8s den Knotenstatus auf Ready: false setzt, was erfolgt, nachdem K8s den Heartbeat eines Knotens verloren hat. K8s hat standardmäßig eine Wartezeit von 40 Sekunden nach dem letzten Heartbeat, bevor Ready: false gesetzt wird. Diese Karenzzeit kann in den K8s-Bereitstellungsoptionen angepasst werden.

Weitere Konfigurationsoptionen finden Sie unter "Node-Healthcheck-Operator Dokumentation".

Zusätzliche Setup-Informationen

Wenn Trident mit aktiviertem Force-Detach installiert wird, werden automatisch zwei zusätzliche Ressourcen im Trident-Namensraum erstellt, um die Integration mit NHC zu erleichtern: TridentNodeRemediationTemplate (TNRT) und ClusterRole.

TridentNodeRemediationTemplate (TNRT):

Das TNRT dient als Vorlage für den NHC Controller, der TNRT verwendet, um bei Bedarf TNR-Ressourcen zu generieren.

```
apiVersion: trident.netapp.io/v1
kind: TridentNodeRemediationTemplate
metadata:
  name: trident-node-remediation-template
  namespace: trident
spec:
  template:
    spec: {}
```

ClusterRole:

Eine Clusterrolle wird während der Installation ebenfalls hinzugefügt, wenn die erzwungene Trennung aktiviert ist. Dadurch erhält NHC Berechtigungen für TNRs im Trident-Namespace.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  labels:
    rbac.ext-remediation/aggregate-to-ext-remediation: "true"
  name: tridentnoderemediation-access
rules:
- apiGroups:
  - trident.netapp.io
  resources:
  - tridentnoderemediationtemplates
  - tridentnoderemediations
  verbs:
  - get
  - list
  - watch
  - create
  - update
  - patch
  - delete
```

K8s-Cluster-Upgrades und Wartung

Um Failover zu verhindern, pausieren Sie das automatisierte Failover während K8s-Wartungsarbeiten oder -Upgrades, bei denen die Nodes voraussichtlich ausfallen oder neu starten. Sie können das NHC CR (siehe oben) pausieren, indem Sie dessen CR patchen:

```
kubectl patch NodeHealthCheck <cr-name> --patch
'{"spec":{"pauseRequests":["<description-for-reason-of-pause>"]}}' --type=merge
```

Dadurch wird das automatische Failover pausiert. Um das automatische Failover wieder zu aktivieren, entfernen Sie die pauseRequests aus der Spezifikation, nachdem die Wartung abgeschlossen ist.

Einschränkungen

- E/A-Operationen werden nur auf den ausgefallenen Knoten für Volumes verhindert, die von force-detach unterstützt werden. Nur Pods, die Volumes/PVCs verwenden, die von force-detach unterstützt werden, werden automatisch entfernt.
- Automatisches Failover und erzwungenes Trennen werden innerhalb des trident-controller-Pods ausgeführt. Fällt der Knoten, auf dem der trident-controller ausgeführt wird, aus, verzögert sich das automatische Failover, bis K8s den Pod auf einen fehlerfreien Knoten verschoben hat.

Integration kundenspezifischer Lösungen zur Überprüfung des Knotenzustands

Sie können den Node Healthcheck Operator durch alternative Tools zur Erkennung von Knotenausfällen ersetzen, um automatisches Failover auszulösen. Um die Kompatibilität mit dem automatisierten Failover-Mechanismus zu gewährleisten, sollte Ihre individuelle Lösung:

- Erstelle einen TNR, wenn ein Knotenausfall erkannt wird, wobei der Name des ausgefallenen Knotens als TNR-CR-Name verwendet wird.
- Löschen Sie den TNR, wenn der Knoten wiederhergestellt ist und sich der TNR im Status „Succeeded“ befindet.

Sicherheit

Sicherheit

Verwenden Sie die hier aufgeführten Empfehlungen, um sicherzustellen, dass Ihre Trident-Installation sicher ist.

Trident in einem eigenen Namensraum ausführen

Es ist wichtig, Anwendungen, Anwendungsadministratoren, Benutzern und Verwaltungsanwendungen den Zugriff auf Trident-Objektdefinitionen oder die Pods zu verwehren, um eine zuverlässige Speicherung zu gewährleisten und potenziell schädliche Aktivitäten zu verhindern.

Um andere Anwendungen und Benutzer von Trident zu trennen, installieren Sie Trident immer in einem eigenen Kubernetes-Namespace (`trident`). Wenn Trident in einem eigenen Namespace installiert wird, wird sichergestellt, dass nur das Kubernetes-Administrationspersonal Zugriff auf den Trident-Pod und die in den Namespaced-CRD-Objekten gespeicherten Artefakte (wie Backend- und CHAP-Secrets, falls zutreffend) hat. Sie sollten sicherstellen, dass nur Administratoren Zugriff auf den Trident-Namespace und damit auf die `tridentctl` Anwendung haben.

CHAP Authentifizierung mit ONTAP SAN-Backends verwenden

Trident unterstützt CHAP-basierte Authentifizierung für ONTAP SAN-Workloads (unter Verwendung der `ontap-san` und `ontap-san-economy` Treiber). NetApp empfiehlt die Verwendung von bidirektionalem CHAP mit Trident für die Authentifizierung zwischen einem Host und dem Storage-Backend.

Für ONTAP-Backends, die die SAN-Speichertreiber verwenden, kann Trident bidirektionales CHAP einrichten und CHAP-Benutzernamen und -Geheimnisse über `tridentctl` verwalten. Siehe "Bereiten Sie die Konfiguration des Backends mit ONTAP SAN-Treibern vor", um zu verstehen, wie Trident CHAP auf ONTAP-

Backends konfiguriert.

Verwenden Sie die CHAP-Authentifizierung mit NetApp HCI und SolidFire Backends

NetApp empfiehlt die Bereitstellung von bidirektionalem CHAP, um die Authentifizierung zwischen einem Host und den NetApp HCI- und SolidFire-Backends sicherzustellen. Trident verwendet ein Secret-Objekt, das zwei CHAP-Passwörter pro Mandant enthält. Wenn Trident installiert ist, verwaltet es die CHAP-Secrets und speichert sie in einem `tridentvolume` CR-Objekt für das jeweilige PV. Wenn Sie ein PV erstellen, verwendet Trident die CHAP-Secrets, um eine iSCSI-Sitzung zu initiieren und über CHAP mit dem NetApp HCI- und SolidFire-System zu kommunizieren.

HINWEIS	Die Volumes, die von Trident erstellt werden, sind keiner Volume-Zugriffsgruppe zugeordnet.

Verwenden Sie Trident mit NVE und NAE

NetApp ONTAP bietet Verschlüsselung ruhender Daten, um sensible Daten im Falle von Diebstahl, Rückgabe oder anderweitiger Verwendung einer Festplatte zu schützen. Weitere Informationen finden Sie unter "Konfigurieren Sie die Übersicht zur NetApp Volume Encryption".

- Wenn NAE im Backend aktiviert ist, wird jedes in Trident bereitgestellte Volume NAE-fähig sein.
  - Sie können das NVE-Verschlüsselungsflag auf `""` setzen, um NAE-fähige Volumes zu erstellen.
- Wenn NAE auf dem Backend nicht aktiviert ist, wird jedes in Trident bereitgestellte Volume NVE-fähig sein, es sei denn, das NVE-Verschlüsselungsflag ist auf `false` (den Standardwert) in der Backend-Konfiguration gesetzt.

HINWEIS	Volumes, die in Trident auf einem NAE-fähigen Backend erstellt wurden, müssen NVE- oder NAE-verschlüsselt sein. • Sie können das NVE-Verschlüsselungsflag auf `true` in der Trident-Backend-Konfiguration setzen, um die NAE-Verschlüsselung zu überschreiben und einen spezifischen Verschlüsselungsschlüssel pro Volume zu verwenden. • Das Setzen des NVE-Verschlüsselungsflags auf `false` einem NAE-fähigen Backend erstellt ein NAE-fähiges Volume. Sie können die NAE-Verschlüsselung nicht deaktivieren, indem Sie das NVE-Verschlüsselungsflag auf `false` setzen.

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

Weitere Informationen zu den Backend-Konfigurationsoptionen finden Sie unter:

- "ONTAP SAN-Konfigurationsoptionen"
- "ONTAP NAS-Konfigurationsoptionen"

Linux Unified Key Setup (LUKS)

Sie können Linux Unified Key Setup (LUKS) aktivieren, um ONTAP SAN- und ONTAP SAN ECONOMY-Volumes auf Trident zu verschlüsseln. Trident unterstützt die Rotation von Passphrasen und die Volume-Erweiterung für LUKS-verschlüsselte Volumes.

In Trident verwenden LUKS-verschlüsselte Volumes die aes-xts-plain64-Verschlüsselung und den Modus, wie empfohlen von "NIST".

HINWEIS	LUKS-Verschlüsselung wird für ASA r2-Systeme nicht unterstützt. Weitere Informationen zu ASA r2-Systemen finden Sie unter "Erfahren Sie mehr über ASA r2-Speichersysteme".

Bevor Sie beginnen

- Auf den Worker-Knoten muss cryptsetup Version 2.1 oder höher (aber niedriger als 3.0) installiert sein. Weitere Informationen finden Sie unter "Gitlab: cryptsetup".
- Aus Leistungsgründen empfiehlt NetApp, dass Worker-Knoten Advanced Encryption Standard New Instructions (AES-NI) unterstützen. Um die AES-NI-Unterstützung zu überprüfen, führen Sie den folgenden Befehl aus:

  ```
  grep "aes" /proc/cpuinfo
  ```

  Wird keine Antwort zurückgegeben, unterstützt Ihr Prozessor kein AES-NI. Weitere Informationen zu AES-NI finden Sie unter: "Intel: Advanced Encryption Standard Instructions (AES-NI)".

LUKS-Verschlüsselung aktivieren

Sie können die volumenbezogene, hostseitige Verschlüsselung mithilfe von Linux Unified Key Setup (LUKS) für ONTAP SAN und ONTAP SAN ECONOMY Volumes aktivieren.

Schritte

1. Definieren Sie die LUKS-Verschlüsselungsattribute in der Backend-Konfiguration. Weitere Informationen zu den Backend-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. Verwenden Sie `parameters.selector` zur Definition der Speicherpools mit LUKS-Verschlüsselung. Beispiel:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: luks
provisioner: csi.trident.netapp.io
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 Datenträger können die Deduplizierung und Komprimierung von ONTAP nicht nutzen.

Backend-Konfiguration für den Import von LUKS-Volumes

Um ein LUKS-Volume zu importieren, müssen Sie `luksEncryption` auf (`true` im Backend setzen. Die `luksEncryption` Option teilt Trident mit, ob das Volume LUKS-kompatibel (`true` ist oder nicht LUKS-kompatibel (`false`, wie im folgenden Beispiel gezeigt.

```
version: 1
storageDriverName: ontap-san
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
svm: trident_svm
username: admin
password: password
defaults:
  luksEncryption: 'true'
  spaceAllocation: 'false'
  snapshotPolicy: default
  snapshotReserve: '10'
```

PVC-Konfiguration für den Import von LUKS-Volumes

Um LUKS-Volumes dynamisch zu importieren, setzen Sie die Annotation `trident.netapp.io/luksEncryption` auf `true` und fügen Sie eine LUKS-fähige Speicherklasse in die PVC ein, wie in diesem Beispiel gezeigt.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: luks-pvc
  namespace: trident
  annotations:
    trident.netapp.io/luksEncryption: "true"
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: luks-sc
```

Eine LUKS-Passphrase rotieren

Sie können die LUKS-Passphrase rotieren und die Rotation bestätigen.

WARNUNG	Vergessen Sie eine Passphrase erst, nachdem Sie überprüft haben, dass sie von keinem Volume, Snapshot oder Secret mehr referenziert wird. Geht eine referenzierte Passphrase verloren, können Sie das Volume möglicherweise nicht einbinden und die Daten bleiben verschlüsselt und unzugänglich.

Über diese Aufgabe

Die Rotation der LUKS-Passphrase erfolgt, wenn ein Pod, der das Volume einbindet, erstellt wird, nachdem eine neue LUKS-Passphrase festgelegt wurde. Wenn ein neuer Pod erstellt wird, vergleicht Trident die LUKS-Passphrase auf dem Volume mit der aktiven Passphrase im Secret.

- Stimmt die Passphrase des Volumes nicht mit der aktiven Passphrase im Secret überein, findet eine Rotation statt.
- Wenn die Passphrase des Volumes mit der aktiven Passphrase im Geheimnis übereinstimmt, wird der `previous-luks-passphrase` Parameter ignoriert.

Schritte

1. Fügen Sie die `node-publish-secret-name` und `node-publish-secret-namespace` StorageClass-Parameter hinzu. 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 Passphrasen auf dem Volume oder Snapshot.

Volumen

```
tridentctl -d get volume luks-pvc1
GET http://127.0.0.1:8000/trident/v1/volume/<volumeID>

...luksPassphraseNames:["A"]
```

Schnappschuss

```
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 neue und die vorherige Passphrase anzugeben. Stellen Sie sicher, dass `previous-luke-passphrase-name` und `previous-luks-passphrase` mit der vorherigen Passphrase übereinstimmen.

```
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 einbindet. Dies ist erforderlich, um die Rotation zu initiieren.

5. Überprüfen Sie, ob die Passphrase geändert wurde.

Volumen

```
tridentctl -d get volume luks-pvc1
GET http://127.0.0.1:8000/trident/v1/volume/<volumeID>

...luksPassphraseNames:["B"]
```

Schnappschuss

```
tridentctl -d get snapshot luks-pvc1
GET http://127.0.0.1:8000/trident/v1/volume/<volumeID>/<snapshotID>

...luksPassphraseNames:["B"]
```

Ergebnisse

Die Passphrase wurde geändert, wenn auf dem Volume und im Snapshot nur die neue Passphrase zurückgegeben wird.

HINWEIS	Wenn zwei Passphrasen zurückgegeben werden, zum Beispiel `luksPassphraseNames: ["B", "A"]`, ist die Rotation unvollständig. Sie können einen neuen Pod auslösen, um zu versuchen, die Rotation abzuschließen.

Volumenerweiterung aktivieren

Sie können die Volumenerweiterung auf einem LUKS-verschlüsselten Volume aktivieren.

Schritte

1. Aktivieren Sie das `CSINodeExpandSecret` Feature-Gate (Beta 1.25+). Siehe "Kubernetes 1.25: Verwendung von Secrets für die knotengesteuerte Erweiterung von CSI-Volumes" für Details.
2. Fügen Sie die `node-expand-secret-name` und `node-expand-secret-namespace` StorageClass-Parameter hinzu. Beispiel:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: luks
provisioner: csi.trident.netapp.io
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, übergibt das kubelet die entsprechenden Anmeldeinformationen an den Treiber.

Kerberos-In-Flight-Verschlüsselung

Durch die Verwendung der Kerberos-In-Flight-Verschlüsselung können Sie die Sicherheit des Datenzugriffs verbessern, indem Sie die Verschlüsselung für den Datenverkehr zwischen Ihrem verwalteten Cluster und dem Storage-Backend aktivieren.

Trident unterstützt Kerberos-Verschlüsselung für ONTAP als Storage-Backend:

- **On-premise ONTAP** - Trident unterstützt die Kerberos-Verschlüsselung über NFSv3- und NFSv4-Verbindungen von Red Hat OpenShift und Upstream-Kubernetes-Clustern zu On-premise ONTAP Volumes.

Sie können Volumes erstellen, löschen, ihre Größe ändern, Snapshots erstellen, klonen, schreibgeschützte Klone erstellen und importieren, die NFS-Verschlüsselung verwenden.

Konfigurieren Sie die Kerberos-Verschlüsselung während der Übertragung mit lokalen ONTAP-Volumes

Sie können die Kerberos-Verschlüsselung für den Speicherdatenverkehr zwischen Ihrem verwalteten Cluster und einem lokalen ONTAP Storage-Backend aktivieren.

HINWEIS	Die Kerberos-Verschlüsselung für NFS-Datenverkehr mit On-Premise ONTAP-Speicher-Backends wird nur mit dem `ontap-nas` storage driver unterstützt.

Bevor Sie beginnen

- Stellen Sie sicher, dass Sie Zugriff auf das `tridentctl` Dienstprogramm haben.
- Stellen Sie sicher, dass Sie über Administratorzugriff auf das ONTAP-Speicher-Backend verfügen.
- Stellen Sie sicher, dass Sie den Namen des oder der Volumes kennen, die Sie vom ONTAP Storage-Backend freigeben werden.
- Stellen Sie sicher, dass Sie die ONTAP Storage-VM für die Unterstützung der Kerberos-Verschlüsselung für NFS-Volumes vorbereitet haben. Siehe "Kerberos auf einem dataLIF aktivieren" für Anweisungen.

- Stellen Sie sicher, dass alle NFSv4-Volumes, die Sie mit Kerberos-Verschlüsselung verwenden, korrekt konfiguriert sind. Siehe den Abschnitt NetApp NFSv4-Domänenkonfiguration (Seite 13) des "NetApp NFSv4-Verbesserungen und Best Practices-Leitfaden".

ONTAP Exportrichtlinien hinzufügen oder ändern

Sie müssen bestehenden ONTAP-Exportrichtlinien Regeln hinzufügen oder neue Exportrichtlinien erstellen, die die Kerberos-Verschlüsselung für das ONTAP Storage-VM-Root-Volume sowie für alle ONTAP-Volumes, die mit dem Upstream-Kubernetes-Cluster gemeinsam genutzt werden, unterstützen. Die Exportrichtlinienregeln, die Sie hinzufügen, oder neuen Exportrichtlinien, die Sie erstellen, müssen die folgenden Zugriffsprotokolle und Zugriffsberechtigungen unterstützen:

Zugriffsprotokolle

Konfigurieren Sie die Exportrichtlinie mit den Zugriffsprotokollen NFS, NFSv3 und NFSv4.

Zugangsdaten

Je nach Ihren Anforderungen an das Volume können Sie eine von drei verschiedenen Versionen der Kerberos-Verschlüsselung konfigurieren:

- **Kerberos 5** - (Authentifizierung und Verschlüsselung)
- **Kerberos 5i** - (Authentifizierung und Verschlüsselung mit Identitätsschutz)
- **Kerberos 5p** - (Authentifizierung und Verschlüsselung mit Identitäts- und Privatsphärenschutz)

Konfigurieren Sie die ONTAP-Exportrichtlinienregel mit den entsprechenden Zugriffsberechtigungen. Wenn Cluster beispielsweise die NFS-Volumes mit einer Mischung aus Kerberos 5i und Kerberos 5p Verschlüsselung einbinden, verwenden Sie die folgenden Zugriffseinstellungen:

Typ	Nur-Lese-Zugriff	Lese-/Schreibzugriff	Superuser-Zugriff
UNIX	Aktiviert	Aktiviert	Aktiviert
Kerberos 5i	Aktiviert	Aktiviert	Aktiviert
Kerberos 5p	Aktiviert	Aktiviert	Aktiviert

Siehe die folgende Dokumentation, um zu erfahren, wie Sie ONTAP Exportrichtlinien und Exportrichtlinienregeln erstellen:

- "Erstellen Sie eine Exportrichtlinie"
- "Fügen Sie einer Exportrichtlinie eine Regel hinzu"

Erstellen Sie ein Storage-Backend

Sie können eine Trident-Speicher-Backend-Konfiguration erstellen, die die Kerberos-Verschlüsselungsfunktion umfasst.

Über diese Aufgabe

Wenn Sie eine Storage-Backend-Konfigurationsdatei erstellen, die die Kerberos-Verschlüsselung konfiguriert, können Sie mit dem `spec.nfsMountOptions`-Parameter eine von drei verschiedenen Versionen der Kerberos-Verschlüsselung angeben:

- `spec.nfsMountOptions: sec=krb5` (Authentifizierung und Verschlüsselung)
- `spec.nfsMountOptions: sec=krb5i` (Authentifizierung und Verschlüsselung mit Identitätsschutz)

- `spec.nfsMountOptions: sec=krb5p` (Authentifizierung und Verschlüsselung mit Identitäts- und Datenschutz)

Geben Sie nur eine Kerberos-Verschlüsselungsstufe an. Wenn Sie in der Parameterliste mehr als eine Kerberos-Verschlüsselungsstufe angeben, wird nur die erste Option verwendet.

Schritte

1. Erstellen Sie auf dem verwalteten Cluster eine Speicher-Backend-Konfigurationsdatei anhand des folgenden Beispiels. Ersetzen Sie Werte in Klammern <> mit Informationen aus Ihrer Umgebung:

```
apiVersion: v1
kind: Secret
metadata:
  name: backend-ontap-nas-secret
type: Opaque
stringData:
  clientID: <CLIENT_ID>
  clientSecret: <CLIENT_SECRET>
---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-ontap-nas
spec:
  version: 1
  storageDriverName: "ontap-nas"
  managementLIF: <STORAGE_VM_MGMT_LIF_IP_ADDRESS>
  dataLIF: <PROTOCOL_LIF_FQDN_OR_IP_ADDRESS>
  svm: <STORAGE_VM_NAME>
  username: <STORAGE_VM_USERNAME_CREDENTIAL>
  password: <STORAGE_VM_PASSWORD_CREDENTIAL>
  nasType: nfs
  nfsMountOptions: ["sec=krb5i"] #can be krb5, krb5i, or krb5p
  qtreesPerFlexvol:
  credentials:
    name: backend-ontap-nas-secret
```

2. Verwenden Sie die Konfigurationsdatei, die Sie im vorherigen Schritt erstellt haben, um das Backend zu erstellen:

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

Schlägt die Backend-Erstellung fehl, liegt ein Fehler in der Backend-Konfiguration vor. Sie können die Protokolle einsehen, um die Ursache zu ermitteln, indem Sie den folgenden Befehl ausführen:

```
tridentctl logs
```

Nachdem Sie das Problem mit der Konfigurationsdatei identifiziert und behoben haben, können Sie den create-Befehl erneut ausführen.

Erstellen Sie eine Speicherklasse

Sie können eine Speicherklasse erstellen, um Volumes mit Kerberos-Verschlüsselung bereitzustellen.

Über diese Aufgabe

Wenn Sie ein Speicherklassenobjekt erstellen, können Sie mit dem `mountOptions`-Parameter eine von drei verschiedenen Versionen der Kerberos-Verschlüsselung angeben:

- `mountOptions: sec=krb5` (Authentifizierung und Verschlüsselung)
- `mountOptions: sec=krb5i` (Authentifizierung und Verschlüsselung mit Identitätsschutz)
- `mountOptions: sec=krb5p` (Authentifizierung und Verschlüsselung mit Identitäts- und Datenschutz)

Geben Sie nur eine Kerberos-Verschlüsselungsstufe an. Wenn Sie in der Parameterliste mehr als eine Kerberos-Verschlüsselungsstufe angeben, wird nur die erste Option verwendet. Wenn die von Ihnen in der Speicher-Backend-Konfiguration angegebene Verschlüsselungsstufe von der Stufe abweicht, die Sie im Speicherklassenobjekt angeben, hat das Speicherklassenobjekt Vorrang.

Schritte

1. Erstellen Sie ein StorageClass Kubernetes-Objekt anhand des folgenden Beispiels:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-nas-sc
provisioner: csi.trident.netapp.io
mountOptions:
  - sec=krb5i #can be krb5, krb5i, or krb5p
parameters:
  backendType: ontap-nas
  storagePools: ontapnas_pool
  trident.netapp.io/nasType: nfs
allowVolumeExpansion: true
```

2. Erstellen Sie die Speicherklasse:

```
kubectl create -f sample-input/storage-class-ontap-nas-sc.yaml
```

3. Stellen Sie sicher, dass die Speicherklasse erstellt wurde:

```
kubectl get sc ontap-nas-sc
```

Sie sollten eine Ausgabe ähnlich der folgenden sehen:

```
NAME          PROVISIONER             AGE
ontap-nas-sc    csi.trident.netapp.io   15h
```

Volumes bereitstellen

Nachdem Sie ein Speicher-Backend und eine Speicherklasse erstellt haben, können Sie nun ein Volume bereitstellen. Anweisungen finden Sie unter "Ein Volume bereitstellen".

Konfigurieren der Kerberos-Verschlüsselung während der Übertragung mit Azure NetApp Files-Volumes

Sie können die Kerberos-Verschlüsselung für den Speicherdatenverkehr zwischen Ihrem verwalteten Cluster und einem einzelnen Azure NetApp Files-Speicher-Backend oder einem virtuellen Pool von Azure NetApp Files-Speicher-Backends aktivieren.

Bevor Sie beginnen

- Stellen Sie sicher, dass Sie Trident auf dem verwalteten Red Hat OpenShift Cluster aktiviert haben.
- Stellen Sie sicher, dass Sie Zugriff auf das `tridentctl` Dienstprogramm haben.
- Stellen Sie sicher, dass Sie das Azure NetApp Files Storage-Backend für die Kerberos-Verschlüsselung vorbereitet haben, indem Sie die Anforderungen beachten und den Anweisungen in "Azure NetApp Files-Dokumentation"folgen.
- Stellen Sie sicher, dass alle NFSv4-Volumes, die Sie mit Kerberos-Verschlüsselung verwenden, korrekt konfiguriert sind. Siehe den Abschnitt NetApp NFSv4-Domänenkonfiguration (Seite 13) des "NetApp NFSv4-Verbesserungen und Best Practices-Leitfaden".

Erstellen Sie ein Storage-Backend

Sie können eine Azure NetApp Files-Speicher-Backend-Konfiguration erstellen, die die Kerberos-Verschlüsselungsfunktion beinhaltet.

Über diese Aufgabe

Wenn Sie eine Konfigurationsdatei für das Storage-Backend erstellen, die die Kerberos-Verschlüsselung konfiguriert, können Sie festlegen, dass sie auf einer von zwei möglichen Ebenen angewendet werden soll:

- Die **Speicher-Backend-Ebene** unter Verwendung des `spec.kerberos` Felds
- Der **virtuelle Poolpegel** unter Verwendung des `spec.storage.kerberos` Feldes

Wenn Sie die Konfiguration auf Ebene des virtuellen Pools definieren, wird der Pool anhand der Bezeichnung in der Speicherklasse ausgewählt.

Auf beiden Ebenen können Sie eine von drei verschiedenen Versionen der Kerberos-Verschlüsselung angeben:

- `kerberos: sec=krb5` (Authentifizierung und Verschlüsselung)

- `kerberos: sec=krb5i` (Authentifizierung und Verschlüsselung mit Identitätsschutz)
- `kerberos: sec=krb5p` (Authentifizierung und Verschlüsselung mit Identitäts- und Datenschutz)

Schritte

1. Erstellen Sie auf dem verwalteten Cluster eine Storage-Backend-Konfigurationsdatei anhand eines der folgenden Beispiele, je nachdem, wo Sie das Storage-Backend definieren müssen (Storage-Backend-Ebene oder virtuelle Pool-Ebene). Ersetzen Sie Werte in Klammern <> mit Informationen aus Ihrer Umgebung:

Beispiel auf Storage-Backend-Ebene

```
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-secret
type: Opaque
stringData:
  clientID: <CLIENT_ID>
  clientSecret: <CLIENT_SECRET>

---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc
spec:
  version: 1
  storageDriverName: azure-netapp-files
  subscriptionID: <SUBSCRIPTION_ID>
  tenantID: <TENANT_ID>
  location: <AZURE_REGION_LOCATION>
  serviceLevel: Standard
  networkFeatures: Standard
  capacityPools: <CAPACITY_POOL>
  resourceGroups: <RESOURCE_GROUP>
  netappAccounts: <NETAPP_ACCOUNT>
  virtualNetwork: <VIRTUAL_NETWORK>
  subnet: <SUBNET>
  nasType: nfs
  kerberos: sec=krb5i #can be krb5, krb5i, or krb5p
  credentials:
    name: backend-tbc-secret
```

Beispiel auf virtueller Pool-Ebene

```
---
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-secret
type: Opaque
stringData:
  clientID: <CLIENT_ID>
  clientSecret: <CLIENT_SECRET>

---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc
spec:
  version: 1
  storageDriverName: azure-netapp-files
  subscriptionID: <SUBSCRIPTION_ID>
  tenantID: <TENANT_ID>
  location: <AZURE_REGION_LOCATION>
  serviceLevel: Standard
  networkFeatures: Standard
  capacityPools: <CAPACITY_POOL>
  resourceGroups: <RESOURCE_GROUP>
  netappAccounts: <NETAPP_ACCOUNT>
  virtualNetwork: <VIRTUAL_NETWORK>
  subnet: <SUBNET>
  nasType: nfs
  storage:
    - labels:
        type: encryption
      kerberos: sec=krb5i #can be krb5, krb5i, or krb5p
  credentials:
    name: backend-tbc-secret
```

2. Verwenden Sie die Konfigurationsdatei, die Sie im vorherigen Schritt erstellt haben, um das Backend zu erstellen:

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

Schlägt die Backend-Erstellung fehl, liegt ein Fehler in der Backend-Konfiguration vor. Sie können die Protokolle einsehen, um die Ursache zu ermitteln, indem Sie den folgenden Befehl ausführen:

```
tridentctl logs
```

Nachdem Sie das Problem mit der Konfigurationsdatei identifiziert und behoben haben, können Sie den create-Befehl erneut ausführen.

Erstellen Sie eine Speicherklasse

Sie können eine Speicherklasse erstellen, um Volumes mit Kerberos-Verschlüsselung bereitzustellen.

Schritte

1. Erstellen Sie ein StorageClass Kubernetes-Objekt anhand des folgenden Beispiels:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: sc-nfs
provisioner: csi.trident.netapp.io
parameters:
  backendType: azure-netapp-files
  trident.netapp.io/nasType: nfs
  selector: type=encryption
```

2. Erstellen Sie die Speicherklasse:

```
kubectl create -f sample-input/storage-class-sc-nfs.yaml
```

3. Stellen Sie sicher, dass die Speicherklasse erstellt wurde:

```
kubectl get sc -sc-nfs
```

Sie sollten eine Ausgabe ähnlich der folgenden sehen:

```
NAME          PROVISIONER             AGE
sc-nfs        csi.trident.netapp.io   15h
```

Volumes bereitstellen

Nachdem Sie ein Speicher-Backend und eine Speicherklasse erstellt haben, können Sie nun ein Volume bereitstellen. Anweisungen finden Sie unter "Ein Volume bereitstellen".

Schützen Sie Anwendungen mit Trident Protect

Erfahren Sie mehr über Trident Protect

NetApp Trident Protect bietet erweiterte Funktionen für das Management von Anwendungsdaten, die die Funktionalität und Verfügbarkeit zustandsbehafteter Kubernetes-Anwendungen verbessern, die durch NetApp ONTAP-Speichersysteme und den NetApp Trident CSI Storage Provisioner unterstützt werden. Trident Protect vereinfacht die Verwaltung, den Schutz und die Migration containerisierter Workloads zwischen Public Clouds und On-Premises-Umgebungen. Es bietet außerdem Automatisierungsfunktionen über seine API und CLI.

Sie können Anwendungen mit Trident Protect schützen, indem Sie benutzerdefinierte Ressourcen (CRs) erstellen oder die Trident Protect CLI verwenden.

Was kommt als Nächstes?

Sie können sich über die Anforderungen von Trident Protect informieren, bevor Sie es installieren:

- "Trident Protect Anforderungen"

Installieren Sie Trident Protect

Trident Protect Anforderungen

Beginnen Sie mit der Überprüfung der Einsatzbereitschaft Ihrer Betriebsumgebung, Anwendungscluster, Anwendungen und Lizenzen. Stellen Sie sicher, dass Ihre Umgebung diese Anforderungen erfüllt, um Trident Protect bereitzustellen und zu betreiben.

Trident Protect Kubernetes-Cluster-Kompatibilität

Trident Protect ist mit einer Vielzahl von vollständig verwalteten und selbstverwalteten Kubernetes-Angeboten kompatibel, einschließlich:

- Amazon Elastic Kubernetes Service (EKS)
- Google Kubernetes Engine (GKE)
- Microsoft Azure Kubernetes Service (AKS)
- Red Hat OpenShift
- SUSE Harvester 1.7.0 (ONTAP iSCSI)
- SUSE Rancher
- VMware Tanzu Portfolio
- Upstream Kubernetes

HINWEIS

- Trident Protect-Backups werden nur auf Linux-Rechenknoten unterstützt. Windows-Rechenknoten werden für Backup-Vorgänge nicht unterstützt.
- Stellen Sie sicher, dass der Cluster, auf dem Sie Trident Protect installieren, mit einem laufenden Snapshot-Controller und den zugehörigen CRDs konfiguriert ist. Informationen zur Installation eines Snapshot-Controllers finden Sie unter "diese Anweisungen".
- Stellen Sie sicher, dass mindestens eine VolumeSnapshotClass existiert. Weitere Informationen finden sich unter "VolumeSnapshotClass".
- Für die Installation von Trident Protect wird Helm 4.x oder höher benötigt.

Trident Protect Speicher-Backend-Kompatibilität

Trident Protect unterstützt die folgenden Storage-Backends:

- Amazon FSx for NetApp ONTAP
- Cloud Volumes ONTAP
- ONTAP Speicherarrays
- Google Cloud NetApp Volumes
- Azure NetApp Files

Stellen Sie sicher, dass Ihr Storage-Backend die folgenden Anforderungen erfüllt:

- Stellen Sie sicher, dass der mit dem Cluster verbundene NetApp Speicher Trident 24.02 oder neuer verwendet (Trident 24.10 wird empfohlen).
- Stellen Sie sicher, dass Sie über ein NetApp ONTAP storage backend verfügen.
- Stellen Sie sicher, dass Sie einen Objektspeicher-Bucket für das Speichern von Backups konfiguriert haben.
- Erstellen Sie alle Anwendungs-Namespaces, die Sie für Anwendungen oder Anwendungsdatenverwaltungsoperationen verwenden möchten. Trident Protect erstellt diese Namespaces nicht für Sie; wenn Sie in einer benutzerdefinierten Ressource einen nicht vorhandenen Namespace angeben, schlägt der Vorgang fehl.

Anforderungen für nas-economy Volumes

Trident Protect unterstützt Backup- und Wiederherstellungsvorgänge auf nas-economy Volumes. Snapshots, Klone und SnapMirror Replikation auf nas-economy Volumes werden derzeit nicht unterstützt. Sie müssen für jedes nas-economy Volume, das Sie mit Trident Protect verwenden möchten, ein Snapshot-Verzeichnis aktivieren.

HINWEIS

Einige Anwendungen sind nicht mit Volumes kompatibel, die ein Snapshot-Verzeichnis verwenden. Für diese Anwendungen müssen Sie das Snapshot-Verzeichnis ausblenden, indem Sie den folgenden Befehl auf dem ONTAP Storage-System ausführen:

```
nfs modify -vserver <svm> -v3-hide-snapshot enabled
```

Sie können das Snapshot-Verzeichnis aktivieren, indem Sie für jedes nas-economy-Volume den folgenden Befehl ausführen und dabei `<volume-UUID>` durch die UUID des Volumes ersetzen, das Sie ändern

möchten:

```
tridentctl update volume <volume-UUID> --snapshot-dir=true --pool-level
=true -n trident
```

HINWEIS	Sie können Snapshot-Verzeichnisse standardmäßig für neue Volumes aktivieren, indem Sie die Trident-Backend-Konfigurationsoption `snapshotDir` auf `true` setzen. Vorhandene Volumes sind davon nicht betroffen.

Schutz von Daten mit KubeVirt VMs

Trident Protect bietet Funktionen zum Einfrieren und Auftauen des Dateisystems für KubeVirt virtuelle Maschinen während Datensicherungsoperationen, um die Datenkonsistenz sicherzustellen. Die Konfigurationsmethode und das Standardverhalten für VM-Einfrieroperationen variieren je nach Trident Protect-Version, wobei neuere Versionen eine vereinfachte Konfiguration über Helm-Chart-Parameter bieten.

HINWEIS	Während Wiederherstellungsvorgängen werden alle `VirtualMachineSnapshots` für eine virtuelle Maschine (VM) erstellten Dateien nicht wiederhergestellt.

Trident Protect 25.10 und neuer

Trident Protect friert KubeVirt-Dateisysteme während der Datensicherungsoperationen automatisch ein und taut sie wieder auf, um Konsistenz zu gewährleisten. Ab Trident Protect 25.10 können Sie dieses Verhalten mit dem `vm.freeze` Parameter während der Helm-Chart-Installation deaktivieren. Der Parameter ist standardmäßig aktiviert.

```
helm install ... --set vm.freeze=false ...
```

Trident Protect 24.10.1 bis 25.06

Ab Trident Protect 24.10.1 friert Trident Protect KubeVirt-Dateisysteme während Datensicherungsoperationen automatisch ein und taut sie wieder auf. Optional können Sie dieses automatische Verhalten mit dem folgenden Befehl deaktivieren:

```
kubectl set env deployment/trident-protect-controller-manager
NEPTUNE_VM_FREEZE=false -n trident-protect
```

Trident Protect 24.10

Trident Protect 24.10 stellt den konsistenten Zustand der KubeVirt VM-Dateisysteme während Datensicherungsoperationen nicht automatisch sicher. Wenn Sie Ihre KubeVirt VM-Daten mit Trident Protect 24.10 schützen möchten, müssen Sie die Funktion zum Einfrieren/Auftauen der Dateisysteme vor der Datensicherungsoperation manuell aktivieren. Dadurch wird sichergestellt, dass sich die Dateisysteme in einem konsistenten Zustand befinden.

Sie können Trident Protect 24.10 so konfigurieren, dass das Einfrieren und Auftauen des VM-Dateisystems während Datensicherungsoperationen verwaltet wird, indem "Virtualisierung konfigurieren" und anschließend den folgenden Befehl verwenden:

```
kubectl set env deployment/trident-protect-controller-manager
NEPTUNE_VM_FREEZE=true -n trident-protect
```

Anforderungen für die SnapMirror-Replikation

NetApp SnapMirror Replikation ist für die Verwendung mit Trident Protect für die folgenden ONTAP Lösungen verfügbar:

- Lokale NetApp FAS, AFF und ASA Systeme. SnapMirror-Replikation mit Trident protect wird derzeit für ASA r2 Systeme nicht unterstützt.
- NetApp ONTAP Select
- NetApp Cloud Volumes ONTAP
- Amazon FSx for NetApp ONTAP

ONTAP Clusteranforderungen für SnapMirror Replikation

Stellen Sie sicher, dass Ihr ONTAP Cluster die folgenden Anforderungen erfüllt, wenn Sie die SnapMirror Replikation nutzen möchten:

- **NetApp Trident**: NetApp Trident muss sowohl auf dem Quell- als auch auf dem Ziel-Kubernetes-Cluster vorhanden sein, die ONTAP als Backend verwenden. Trident Protect unterstützt die Replikation mit NetApp SnapMirror-Technologie unter Verwendung von Speicherklassen, die von den folgenden Treibern unterstützt werden:
  - `ontap-nas`: NFS
  - `ontap-san`: iSCSI
  - `ontap-san`: FC
  - `ontap-san`: NVMe/TCP (erfordert mindestens ONTAP Version 9.15.1)
- **Lizenzen**: ONTAP SnapMirror asynchrone Lizenzen mit dem Data Protection Bundle müssen sowohl auf dem Quell- als auch auf dem Ziel-ONTAP-Cluster aktiviert sein. Weitere Informationen sind unter "SnapMirror Lizenzierungsübersicht in ONTAP" verfügbar.

  Ab ONTAP 9.10.1 werden alle Lizenzen als NetApp Lizenzdatei (NLF) bereitgestellt, bei der es sich um eine einzelne Datei handelt, die mehrere Funktionen aktiviert. Weitere Informationen sind unter "In ONTAP One enthaltene Lizenzen" verfügbar.

HINWEIS	Es wird ausschließlich SnapMirror asynchroner Schutz unterstützt.

Peering-Überlegungen für die SnapMirror Replikation

Stellen Sie sicher, dass Ihre Umgebung die folgenden Anforderungen erfüllt, wenn Sie Storage-Backend-Peering verwenden möchten:

- **Cluster und SVM**: Die ONTAP-Speicher-Backends müssen per Peering verbunden sein. Weitere Informationen sind unter "Cluster- und SVM-Peering-Übersicht" verfügbar.

HINWEIS	Stellen Sie sicher, dass die in der Replikationsbeziehung zwischen zwei ONTAP Clustern verwendeten SVM-Namen eindeutig sind.

- **NetApp Trident und SVM**: Die verbundenen Remote-SVMs müssen für NetApp Trident auf dem Ziel-Cluster verfügbar sein.
- **Verwaltete Backends**: Sie müssen ONTAP-Speicher-Backends in Trident Protect hinzufügen und verwalten, um eine Replikationsbeziehung zu erstellen.

Trident / ONTAP-Konfiguration für SnapMirror-Replikation

Trident Protect erfordert, dass Sie mindestens ein Storage-Backend konfigurieren, das die Replikation sowohl für den Quell- als auch für den Ziel-Cluster unterstützt. Wenn der Quell- und der Ziel-Cluster identisch sind, sollte die Zielanwendung für die beste Ausfallsicherheit ein anderes Storage-Backend als die Quellanwendung verwenden.

Kubernetes-Cluster-Anforderungen für SnapMirror Replikation

Stellen Sie sicher, dass Ihre Kubernetes-Cluster die folgenden Anforderungen erfüllen:

- **AppVault-Zugänglichkeit**: Sowohl Quell- als auch Ziel-Cluster müssen Netzwerkzugriff haben, um vom AppVault zu lesen und darauf zu schreiben, damit Anwendungsobjekte repliziert werden können.
- **Netzwerkanbindung**: Konfigurieren Sie Firewall-Regeln, Bucket-Berechtigungen und IP-Zulassungslisten, um die Kommunikation zwischen beiden Clustern und dem AppVault über WANs hinweg zu ermöglichen.

HINWEIS	Viele Unternehmensumgebungen setzen strenge Firewall-Richtlinien für WAN-Verbindungen ein. Klären Sie diese Netzwerkanforderungen mit Ihrem Infrastrukturteam, bevor Sie die Replikation konfigurieren.

Trident Protect installieren und konfigurieren

Wenn Ihre Umgebung die Anforderungen für Trident Protect erfüllt, können Sie Trident Protect mithilfe der folgenden Schritte auf Ihrem Cluster installieren. Sie können Trident Protect von NetApp beziehen oder es aus Ihrer eigenen privaten Registry installieren. Die Installation aus einer privaten Registry ist hilfreich, wenn Ihr Cluster keinen Internetzugang hat.

Installieren Sie Trident Protect

Installieren Sie Trident Protect von NetApp

Schritte

1. Fügen Sie das Trident Helm repository hinzu:

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

2. Verwenden Sie Helm, um Trident Protect zu installieren. Ersetzen Sie `<name-of-cluster>` durch einen Clusternamen, der dem Cluster zugewiesen und zur Identifizierung der Backups und Snapshots des Clusters verwendet wird:

```
helm install trident-protect netapp-trident-protect/trident-protect
--set clusterName=<name-of-cluster> --version 100.2602.0 --create
-namespace --namespace trident-protect
```

3. Optional können Sie zur Aktivierung der Debug-Protokollierung (empfohlen zur Fehlerbehebung) Folgendes verwenden:

```
helm install trident-protect netapp-trident-protect/trident-protect
--set clusterName=<name-of-cluster> --set logLevel=debug --version
100.2602.0 --create-namespace --namespace trident-protect
```

   Die Debug-Protokollierung hilft dem NetApp Support, Probleme zu beheben, ohne dass Änderungen des Protokollierungslevels oder eine Reproduktion des Problems erforderlich sind.

Installieren Sie Trident Protect aus einem privaten Registry.

Sie können Trident Protect aus einer privaten Image-Registry installieren, wenn Ihr Kubernetes-Cluster keinen Internetzugang hat. Ersetzen Sie in diesen Beispielen die Werte in Klammern durch Informationen aus Ihrer Umgebung:

Schritte

1. Laden Sie die folgenden Images auf Ihre lokale Maschine herunter, aktualisieren Sie die Tags und laden Sie sie anschließend in Ihre private Registry hoch:

```
docker.io/netapp/controller:26.02.0
docker.io/netapp/restic:26.02.0
docker.io/netapp/kopia:26.02.0
docker.io/netapp/kopiablockrestore:26.02.0
docker.io/netapp/trident-autosupport:26.02.0
docker.io/netapp/exechook:26.02.0
docker.io/netapp/resourcebackup:26.02.0
docker.io/netapp/resourcerestore:26.02.0
docker.io/netapp/resourcedelete:26.02.0
docker.io/netapp/trident-protect-utils:v1.0.0
```

Beispiel:

```
docker pull docker.io/netapp/controller:26.02.0
```

```
docker tag docker.io/netapp/controller:26.02.0 <private-registry-url>/controller:26.02.0
```

```
docker push <private-registry-url>/controller:26.02.0
```

HINWEIS Um das Helm-Chart zu erhalten, laden Sie zunächst das Helm-Chart auf einem Rechner mit Internetzugang mit `helm pull trident-protect --version 100.2602.0 --repo https://netapp.github.io/trident-protect-helm-chart` herunter, kopieren Sie dann die resultierende `trident-protect-100.2602.0.tgz` Datei in Ihre Offline-Umgebung und installieren Sie es mit `helm install trident-protect ./trident-protect-100.2602.0.tgz` anstelle der Repository-Referenz im letzten Schritt.

2. Erstellen Sie den Trident Protect-Systemnamensraum:

```
kubectl create ns trident-protect
```

3. Melden Sie sich bei der Registry an:

```
helm registry login <private-registry-url> -u <account-id> -p <api-token>
```

4. Erstellen Sie ein Pull-Secret zur Verwendung für die private Registry-Authentifizierung:

```
kubectl create secret docker-registry regcred --docker
-username=<registry-username> --docker-password=<api-token> -n
trident-protect --docker-server=<private-registry-url>
```

5. Fügen Sie das Trident Helm repository hinzu:

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

6. Erstellen Sie eine Datei mit dem Namen `protectValues.yaml`. Stellen Sie sicher, dass sie die folgenden Trident Protect-Einstellungen enthält:

```
---
imageRegistry: <private-registry-url>
imagePullSecrets:
  - name: regcred
```

HINWEIS

Die `imageRegistry` und `imagePullSecrets` Werte gelten für alle Komponentenbilder, einschließlich `resourcebackup` und `resourcerestore`. Wenn Sie Bilder in einen bestimmten Repository-Pfad innerhalb Ihrer Registry hochladen (zum Beispiel `example.com:443/my-repo`), geben Sie den vollständigen Pfad im Registry-Feld an. Dadurch wird sichergestellt, dass alle Bilder aus `<private-registry-url>/<image-name>:<tag>` abgerufen werden.

7. Verwenden Sie Helm, um Trident Protect zu installieren. Ersetzen Sie `<name_of_cluster>` durch einen Clusternamen, der dem Cluster zugewiesen und zur Identifizierung der Backups und Snapshots des Clusters verwendet wird:

```
helm install trident-protect netapp-trident-protect/trident-protect
--set clusterName=<name_of_cluster> --version 100.2602.0 --create
-namespace --namespace trident-protect -f protectValues.yaml
```

8. Optional können Sie zur Aktivierung der Debug-Protokollierung (empfohlen zur Fehlerbehebung) Folgendes verwenden:

```
helm install trident-protect netapp-trident-protect/trident-protect
--set clusterName=<name-of-cluster> --set logLevel=debug --version
100.2602.0 --create-namespace --namespace trident-protect -f
protectValues.yaml
```

Die Debug-Protokollierung hilft dem NetApp Support, Probleme zu beheben, ohne dass Änderungen des Protokollierungslevels oder eine Reproduktion des Problems erforderlich sind.

HINWEIS	Weitere Helm-Chart-Konfigurationsoptionen, einschließlich AutoSupport-Einstellungen und Namespace-Filterung, finden Sie unter "Trident Protect-Installation anpassen".

Installieren Sie das Trident Protect CLI-Plugin

Sie können das Trident Protect Befehlszeilen-Plugin verwenden, eine Erweiterung des Trident `tridentctl`-Dienstprogramms, um Trident Protect benutzerdefinierte Ressourcen (CRs) zu erstellen und mit ihnen zu interagieren.

Installieren Sie das Trident Protect CLI-Plugin

Bevor Sie das Befehlszeilenprogramm verwenden, müssen Sie es auf dem Rechner installieren, mit dem Sie auf Ihren Cluster zugreifen. Führen Sie die folgenden Schritte aus, je nachdem, ob Ihr Rechner eine x64- oder ARM-CPU verwendet.

Plugin für Linux AMD64-CPUs herunterladen

Schritte

1. Laden Sie das Trident Protect CLI-Plugin herunter:

```
curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-protect/releases/download/26.02.0/tridentctl-protect-linux-amd64
```

Plugin für Linux ARM64 CPUs herunterladen

Schritte

1. Laden Sie das Trident Protect CLI-Plugin herunter:

```
curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-protect/releases/download/26.02.0/tridentctl-protect-linux-arm64
```

Plugin für Mac AMD64-CPUs herunterladen

Schritte

1. Laden Sie das Trident Protect CLI-Plugin herunter:

```
curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-protect/releases/download/26.02.0/tridentctl-protect-macos-amd64
```

Plugin für Mac ARM64 CPUs herunterladen

Schritte

1. Laden Sie das Trident Protect CLI-Plugin herunter:

```
curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-protect/releases/download/26.02.0/tridentctl-protect-macos-arm64
```

1. Aktivieren Sie die Ausführungsberechtigungen für die Plugin-Binärdatei:

```
chmod +x tridentctl-protect
```

2. Kopieren Sie die Plugin-Binärdatei in ein Verzeichnis, das in Ihrer PATH-Variablen definiert ist. Zum Beispiel, `/usr/bin` oder `/usr/local/bin` (möglicherweise benötigen Sie erhöhte Berechtigungen):

```
cp ./tridentctl-protect /usr/local/bin/
```

3. Optional können Sie die Plugin-Binärdatei in ein Verzeichnis in Ihrem Home-Verzeichnis kopieren. In diesem Fall wird empfohlen, sicherzustellen, dass sich das Verzeichnis in Ihrer PATH-Variablen befindet:

```
cp ./tridentctl-protect ~/bin/
```

HINWEIS Durch das Kopieren des Plugins an einen Ort in Ihrer PATH-Variablen können Sie das Plugin verwenden, indem Sie `tridentctl-protect` oder `tridentctl protect` von jedem beliebigen Ort aus eingeben.

Hilfe zum Trident CLI-Plugin anzeigen

Sie können die integrierten Plugin-Hilfefunktionen verwenden, um detaillierte Hilfe zu den Funktionen des Plugins zu erhalten:

Schritte

1. Verwenden Sie die Hilfefunktion, um Nutzungshinweise anzuzeigen:

```
tridentctl-protect help
```

Automatische Befehlsvervollständigung aktivieren

Nachdem Sie das Trident Protect CLI-Plugin installiert haben, können Sie die automatische Vervollständigung für bestimmte Befehle aktivieren.

Aktivieren Sie die automatische Vervollständigung für die Bash-Shell

Schritte

1. Erstellen Sie das Vervollständigungsskript:

```
tridentctl-protect completion bash > tridentctl-completion.bash
```

2. Erstellen Sie ein neues Verzeichnis in Ihrem Home-Verzeichnis, um das Skript darin zu speichern:

```
mkdir -p ~/.bash/completions
```

3. Verschieben Sie das heruntergeladene Skript in das `~/.bash/completions` Verzeichnis:

```
mv tridentctl-completion.bash ~/.bash/completions/
```

4. Fügen Sie die folgende Zeile in die `~/.bashrc` Datei in Ihrem Home-Verzeichnis ein:

```
source ~/.bash/completions/tridentctl-completion.bash
```

Automatische Vervollständigung für die Z shell aktivieren

Schritte

1. Erstellen Sie das Vervollständigungsskript:

```
tridentctl-protect completion zsh > tridentctl-completion.zsh
```

2. Erstellen Sie ein neues Verzeichnis in Ihrem Home-Verzeichnis, um das Skript darin zu speichern:

```
mkdir -p ~/.zsh/completions
```

3. Verschieben Sie das heruntergeladene Skript in das `~/.zsh/completions` Verzeichnis:

```
mv tridentctl-completion.zsh ~/.zsh/completions/
```

4. Fügen Sie die folgende Zeile in die `~/.zprofile` Datei in Ihrem Home-Verzeichnis ein:

```
source ~/.zsh/completions/tridentctl-completion.zsh
```

Ergebnis

Beim nächsten Login in die Shell können Sie die Befehlsvervollständigung mit dem tridentctl-protect plugin nutzen.

Trident Protect-Installation anpassen

Sie können die Standardkonfiguration von Trident Protect an die spezifischen Anforderungen Ihrer Umgebung anpassen.

Geben Sie die Ressourcenbeschränkungen für den Trident Protect-Container an.

Sie können eine Konfigurationsdatei verwenden, um Ressourcenlimits für Trident Protect-Container festzulegen, nachdem Sie Trident Protect installiert haben. Durch das Festlegen von Ressourcenlimits können Sie steuern, wie viele Ressourcen des Clusters von Trident Protect-Operationen verbraucht werden.

Schritte

1. Erstellen Sie eine Datei mit dem Namen `resourceLimits.yaml`.
2. Füllen Sie die Datei mit Ressourcenlimitoptionen für Trident Protect-Container entsprechend den Anforderungen Ihrer Umgebung.

   Die folgende Beispiel-Konfigurationsdatei zeigt die verfügbaren Einstellungen und enthält die Standardwerte für jede Ressourcenbegrenzung:

```
---
jobResources:
  defaults:
    limits:
      cpu: 8000m
      memory: 10000Mi
      ephemeralStorage: ""
    requests:
      cpu: 100m
      memory: 100Mi
      ephemeralStorage: ""
  resticVolumeBackup:
    limits:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
    requests:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
  resticVolumeRestore:
    limits:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
```

```
    requests:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
  kopiaVolumeBackup:
    limits:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
    requests:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
  kopiaVolumeRestore:
    limits:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
    requests:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
```

3. Wenden Sie die Werte aus der `resourceLimits.yaml` Datei an:

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect -f resourceLimits.yaml --reuse-values
```

Sicherheitskontextbeschränkungen anpassen

Sie können eine Konfigurationsdatei verwenden, um die OpenShift Security Context Constraint (SCCs) für Trident Protect-Container nach der Installation von Trident Protect zu ändern. Diese Beschränkungen definieren Sicherheitsvorgaben für Pods in einem Red Hat OpenShift Cluster.

Schritte

1. Erstellen Sie eine Datei mit dem Namen `sccconfig.yaml`.
2. Fügen Sie die SCC-Option zur Datei hinzu und ändern Sie die Parameter entsprechend den Anforderungen Ihrer Umgebung.

   Das folgende Beispiel zeigt die Standardwerte der Parameter für die SCC-Option:

```
scc:
  create: true
  name: trident-protect-job
  priority: 1
```

Diese Tabelle beschreibt die Parameter für die SCC-Option:

Parameter	Beschreibung	Standard
erstellen	Legt fest, ob eine SCC-Ressource erstellt werden kann. Eine SCC-Ressource wird nur erstellt, wenn `scc.create` auf `true` gesetzt ist und der Helm-Installationsprozess eine OpenShift-Umgebung identifiziert. Wenn nicht auf OpenShift gearbeitet wird oder wenn `scc.create` auf `false` gesetzt ist, wird keine SCC-Ressource erstellt.	true
Name	Gibt den Namen des SCC an.	trident-protect-job
Priorität	Legt die Priorität der SCC fest. SCCs mit höheren Prioritätswerten werden vor solchen mit niedrigeren Werten bewertet.	1

3. Wenden Sie die Werte aus der `sccconfig.yaml` Datei an:

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect -f sccconfig.yaml --reuse-values
```

Dadurch werden die Standardwerte durch die in der `sccconfig.yaml` Datei angegebenen Werte ersetzt.

Konfigurieren Sie zusätzliche Trident Protect Helm-Chart-Einstellungen

Sie können die AutoSupport-Einstellungen und die Namensraumfilterung an Ihre spezifischen Anforderungen anpassen. Die folgende Tabelle beschreibt die verfügbaren Konfigurationsparameter:

Parameter	Typ	Beschreibung
autoSupport.proxy	Zeichenkette	Konfiguriert eine Proxy-URL für NetApp AutoSupport-Verbindungen. Verwenden Sie dies, um Support-Bundle-Uploads über einen Proxyserver zu leiten. Beispiel: `http://my.proxy.url`.

Parameter	Typ	Beschreibung
autoSupport.insecure	boolescher Wert	Überspringt die TLS-Verifizierung für AutoSupport Proxyserver-Verbindungen, wenn auf `true` gesetzt. Nur für unsichere Proxyserver-Verbindungen verwenden. (Standard: `false`)
autoSupport.enabled	boolescher Wert	Aktiviert oder deaktiviert tägliche Trident Protect AutoSupport Bundle-Uploads. Wenn auf `false` gesetzt, werden geplante tägliche Uploads deaktiviert, aber Sie können weiterhin Support-Bundles manuell generieren. (Standard: `true`)
restoreSkipNamespaceAnnotations	Zeichenkette	Kommagetrennte Liste von Namespace-Annotationen, die von Sicherungs- und Wiederherstellungsvorgängen ausgeschlossen werden sollen. Ermöglicht das Filtern von Namespaces anhand von Annotationen.
restoreSkipNamespaceLabels	Zeichenkette	Durch Kommas getrennte Liste von Namespace-Labels, die von Sicherungs- und Wiederherstellungsvorgängen ausgeschlossen werden. Ermöglicht das Filtern von Namespaces anhand von Labels.

Sie können diese Optionen entweder mit einer YAML-Konfigurationsdatei oder Befehlszeilen-Flags konfigurieren:

YAML-Datei verwenden

Schritte

1. Erstelle eine Konfigurationsdatei und benenne sie `values.yaml`.
2. Fügen Sie in der von Ihnen erstellten Datei die Konfigurationsoptionen hinzu, die Sie anpassen möchten.

```
autoSupport:
  enabled: false
  proxy: http://my.proxy.url
  insecure: true
restoreSkipNamespaceAnnotations: "annotation1,annotation2"
restoreSkipNamespaceLabels: "label1,label2"
```

3. Nachdem Sie die `values.yaml` Datei mit den korrekten Werten gefüllt haben, wenden Sie die Konfigurationsdatei an:

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect -f values.yaml --reuse-values
```

CLI-Flag verwenden

Schritte

1. Verwenden Sie den folgenden Befehl mit dem `--set` Flag, um einzelne Parameter anzugeben:

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect \
  --set autoSupport.enabled=false \
  --set autoSupport.proxy=http://my.proxy.url \
  --set-string
restoreSkipNamespaceAnnotations="{annotation1,annotation2}" \
  --set-string restoreSkipNamespaceLabels="{label1,label2}" \
  --reuse-values
```

Trident Protect-Pods auf bestimmte Knoten beschränken

Sie können die Kubernetes nodeSelector Knotenauswahlbeschränkung verwenden, um anhand von Knotenbezeichnungen zu steuern, welche Ihrer Knoten berechtigt sind, Trident Protect-Pods auszuführen. Standardmäßig ist Trident Protect auf Knoten beschränkt, auf denen Linux ausgeführt wird. Sie können diese Beschränkungen je nach Bedarf weiter anpassen.

Schritte

1. Erstellen Sie eine Datei mit dem Namen `nodeSelectorConfig.yaml`.

2. Fügen Sie die nodeSelector-Option zur Datei hinzu und bearbeiten Sie die Datei, um Knotenbezeichnungen hinzuzufügen oder zu ändern, um die Einschränkungen entsprechend den Anforderungen Ihrer Umgebung anzupassen. Die folgende Datei enthält beispielsweise die Standardeinschränkung des Betriebssystems, zielt aber auch auf eine bestimmte Region und einen bestimmten Anwendungsnamen ab:

```
nodeSelector:
  kubernetes.io/os: linux
  region: us-west
  app.kubernetes.io/name: mysql
```

3. Wenden Sie die Werte aus der `nodeSelectorConfig.yaml` Datei an:

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect -f nodeSelectorConfig.yaml --reuse-values
```

Dadurch werden die Standardbeschränkungen durch die von Ihnen in der `nodeSelectorConfig.yaml` Datei angegebenen Beschränkungen ersetzt.

Trident Protect verwalten

Trident Protect-Autorisierung und Zugriffskontrolle verwalten

Trident Protect nutzt das Kubernetes-Modell der rollenbasierten Zugriffssteuerung (RBAC). Standardmäßig stellt Trident Protect einen einzelnen System-Namespace und das zugehörige Standard-Dienstkonto bereit. Wenn Sie eine Organisation mit vielen Benutzern oder spezifischen Sicherheitsanforderungen haben, können Sie die RBAC-Funktionen von Trident Protect nutzen, um den Zugriff auf Ressourcen und Namespaces detaillierter zu steuern.

Der Clusteradministrator hat stets Zugriff auf Ressourcen im `trident-protect` Standard-Namespace und kann auch auf Ressourcen in allen anderen Namespaces zugreifen. Um den Zugriff auf Ressourcen und Anwendungen zu steuern, müssen Sie zusätzliche Namespaces erstellen und Ressourcen und Anwendungen zu diesen Namespaces hinzufügen.

Beachten Sie, dass keine Benutzer Anwendungsdatenverwaltungs-CRs im Standard `trident-protect` -Namespace erstellen können. Sie müssen Anwendungsdatenverwaltungs-CRs in einem Anwendungs-Namespace erstellen (als Best Practice erstellen Sie Anwendungsdatenverwaltungs-CRs im selben Namespace wie die zugehörige Anwendung).

HINWEIS	Nur Administratoren sollten Zugriff auf privilegierte Trident Protect benutzerdefinierte Ressourcenobjekte haben, darunter: • **AppVault**: Erfordert Bucket-Anmeldeinformationen • **AutoSupportBundle**: Erfasst Metriken, Protokolle und andere sensible Trident Protect-Daten • **AutoSupportBundleSchedule**: Verwaltet Protokollerfassungspläne Als bewährte Methode verwenden Sie rollenbasierte Zugriffssteuerung (RBAC), um den Zugriff auf privilegierte Objekte auf Administratoren zu beschränken.

Weitere Informationen darüber, wie RBAC den Zugriff auf Ressourcen und Namensräume regelt, finden Sie unter "Kubernetes RBAC-Dokumentation".

Weitere Informationen zu Servicekonten finden Sie in der "Dokumentation zum Kubernetes Servicekonto".

Beispiel: Zugriff für zwei Gruppen von Benutzern verwalten

Eine Organisation verfügt beispielsweise über einen Cluster-Administrator, eine Gruppe von Engineering-Benutzern und eine Gruppe von Marketing-Benutzern. Der Cluster-Administrator würde die folgenden Aufgaben ausführen, um eine Umgebung zu schaffen, in der die Engineering-Gruppe und die Marketing-Gruppe jeweils nur auf die Ressourcen zugreifen können, die ihren jeweiligen Namensräumen zugewiesen sind.

Schritt 1: Erstellen Sie einen Namensraum, um Ressourcen für jede Gruppe zu enthalten

Durch das Erstellen eines Namensraums können Sie Ressourcen logisch trennen und besser steuern, wer Zugriff auf diese Ressourcen hat.

Schritte

1. Erstellen Sie einen Namensraum für die Engineering-Gruppe:

```
kubectl create ns engineering-ns
```

2. Erstellen Sie einen Namensraum für die Marketinggruppe:

```
kubectl create ns marketing-ns
```

Schritt 2: Erstellen Sie neue Dienstkonten, um mit Ressourcen in jedem Namespace zu interagieren

Jeder neu erstellte Namespace verfügt über ein Standarddienstkonto, aber Sie sollten für jede Benutzergruppe ein Dienstkonto erstellen, damit Sie die Berechtigungen in Zukunft bei Bedarf weiter zwischen den Gruppen aufteilen können.

Schritte

1. Erstellen Sie ein Dienstkonto für die Engineering-Gruppe:

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: eng-user
  namespace: engineering-ns
```

2. Erstellen Sie ein Servicekonto für die Marketinggruppe:

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: mkt-user
  namespace: marketing-ns
```

Schritt 3: Erstellen Sie ein Geheimnis für jedes neue Dienstkonto

Ein Dienstkontogeheimnis wird zur Authentifizierung mit dem Dienstkonto verwendet und kann im Falle einer Kompromittierung leicht gelöscht und neu erstellt werden.

Schritte

1. Erstellen Sie ein Secret für das Engineering-Service-Konto:

```
apiVersion: v1
kind: Secret
metadata:
  annotations:
    kubernetes.io/service-account.name: eng-user
  name: eng-user-secret
  namespace: engineering-ns
type: kubernetes.io/service-account-token
```

2. Erstellen Sie ein Geheimnis für das Marketing-Service-Konto:

```
apiVersion: v1
kind: Secret
metadata:
  annotations:
    kubernetes.io/service-account.name: mkt-user
  name: mkt-user-secret
  namespace: marketing-ns
type: kubernetes.io/service-account-token
```

Schritt 4: Erstellen Sie ein RoleBinding-Objekt, um das ClusterRole-Objekt an jedes neue Dienstkonto zu binden

Ein Standard-ClusterRole-Objekt wird erstellt, wenn Sie Trident Protect installieren. Sie können dieses ClusterRole an das Dienstkonto binden, indem Sie ein RoleBinding-Objekt erstellen und anwenden.

Schritte

1. Binden Sie die ClusterRole an das Engineering-Servicekonto:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: engineering-ns-tenant-rolebinding
  namespace: engineering-ns
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: trident-protect-tenant-cluster-role
subjects:
- kind: ServiceAccount
  name: eng-user
  namespace: engineering-ns
```

2. Binden Sie die ClusterRole an das Marketing-Servicekonto:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: marketing-ns-tenant-rolebinding
  namespace: marketing-ns
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: trident-protect-tenant-cluster-role
subjects:
- kind: ServiceAccount
  name: mkt-user
  namespace: marketing-ns
```

Schritt 5: Berechtigungen testen

Testen Sie, ob die Berechtigungen korrekt sind.

Schritte

1. Bestätigen Sie, dass technische Benutzer auf technische Ressourcen zugreifen können:

```
kubectl auth can-i --as=system:serviceaccount:engineering-ns:eng-user
get applications.protect.trident.netapp.io -n engineering-ns
```

2. Bestätigen Sie, dass technische Benutzer keinen Zugriff auf Marketingressourcen haben:

```
kubectl auth can-i --as=system:serviceaccount:engineering-ns:eng-user
get applications.protect.trident.netapp.io -n marketing-ns
```

Schritt 6: Zugriff auf AppVault-Objekte gewähren

Um Datenverwaltungsaufgaben wie Backups und Snapshots durchzuführen, muss der Clusteradministrator einzelnen Benutzern Zugriff auf AppVault-Objekte gewähren.

Schritte

1. Erstellen und wenden Sie eine AppVault- und Secret-Kombinations-YAML-Datei an, die einem Benutzer Zugriff auf eine AppVault gewährt. Beispielsweise gewährt die folgende CR einem Benutzer Zugriff auf eine AppVault `eng-user`:

```
apiVersion: v1
data:
  accessKeyID: <ID_value>
  secretAccessKey: <key_value>
kind: Secret
metadata:
  name: appvault-for-eng-user-only-secret
  namespace: trident-protect
type: Opaque
---
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: appvault-for-eng-user-only
  namespace: trident-protect # Trident Protect system namespace
spec:
  providerConfig:
    azure:
      accountName: ""
      bucketName: ""
      endpoint: ""
    gcp:
      bucketName: ""
      projectID: ""
    s3:
      bucketName: testbucket
      endpoint: 192.168.0.1:30000
      secure: "false"
      skipCertValidation: "true"
  providerCredentials:
    accessKeyID:
      valueFromSecret:
        key: accessKeyID
        name: appvault-for-eng-user-only-secret
    secretAccessKey:
      valueFromSecret:
        key: secretAccessKey
        name: appvault-for-eng-user-only-secret
  providerType: GenericS3
```

2. Erstellen und wenden Sie eine Role-CR an, um Clusteradministratoren die Möglichkeit zu geben, Zugriff auf bestimmte Ressourcen in einem Namespace zu gewähren. Beispiel:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: eng-user-appvault-reader
  namespace: trident-protect
rules:
- apiGroups:
  - protect.trident.netapp.io
  resourceNames:
  - appvault-for-enguser-only
  resources:
  - appvaults
  verbs:
  - get
```

3. Erstellen und wenden Sie eine RoleBinding CR an, um die Berechtigungen an den Benutzer eng-user zu binden. Beispiel:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: eng-user-read-appvault-binding
  namespace: trident-protect
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: eng-user-appvault-reader
subjects:
- kind: ServiceAccount
  name: eng-user
  namespace: engineering-ns
```

4. Überprüfen Sie, ob die Berechtigungen korrekt sind.

   a. Versuch, AppVault-Objektinformationen für alle Namensräume abzurufen:

   ```
   kubectl get appvaults -n trident-protect
   --as=system:serviceaccount:engineering-ns:eng-user
   ```

   Sie sollten eine Ausgabe ähnlich der folgenden sehen:

```
Error from server (Forbidden): appvaults.protect.trident.netapp.io is
forbidden: User "system:serviceaccount:engineering-ns:eng-user"
cannot list resource "appvaults" in API group
"protect.trident.netapp.io" in the namespace "trident-protect"
```

b. Testen Sie, ob der Benutzer die AppVault-Informationen abrufen kann, auf die er nun Zugriffsberechtigung hat:

```
kubectl auth can-i --as=system:serviceaccount:engineering-ns:eng-user
get appvaults.protect.trident.netapp.io/appvault-for-eng-user-only -n
trident-protect
```

Sie sollten eine Ausgabe ähnlich der folgenden sehen:

```
yes
```

Ergebnis

Die Benutzer, denen Sie AppVault-Berechtigungen erteilt haben, sollten in der Lage sein, autorisierte AppVault-Objekte für Anwendungsdatenverwaltungsoperationen zu verwenden und sollten nicht in der Lage sein, auf Ressourcen außerhalb der zugewiesenen Namensräume zuzugreifen oder neue Ressourcen zu erstellen, auf die sie keinen Zugriff haben.

Überwachen Sie die Trident Protect-Ressourcen

Sie können die Open-Source-Tools kube-state-metrics, Prometheus und Alertmanager verwenden, um die Integrität der durch Trident Protect geschützten Ressourcen zu überwachen.

Der Dienst kube-state-metrics generiert Metriken aus der Kubernetes-API-Kommunikation. Die Verwendung mit Trident Protect liefert nützliche Informationen über den Zustand der Ressourcen in Ihrer Umgebung.

Prometheus ist ein Toolkit, das die von kube-state-metrics generierten Daten erfassen und als leicht lesbare Informationen über diese Objekte darstellen kann. Zusammen bieten kube-state-metrics und Prometheus eine Möglichkeit, den Zustand und Status der Ressourcen zu überwachen, die Sie mit Trident Protect verwalten.

Alertmanager ist ein Dienst, der die von Tools wie Prometheus gesendeten Warnmeldungen aufnimmt und sie an Ziele weiterleitet, die Sie konfigurieren.

HINWEIS

Die in diesen Schritten enthaltenen Konfigurationen und Anleitungen sind lediglich Beispiele; Sie müssen sie an Ihre Umgebung anpassen. Spezifische Anweisungen und Unterstützung finden Sie in der folgenden offiziellen Dokumentation:

- "kube-state-metrics Dokumentation"
- "Prometheus-Dokumentation"
- "Alertmanager-Dokumentation"

Schritt 1: Installieren Sie die Überwachungstools

Um die Ressourcenüberwachung in Trident Protect zu aktivieren, müssen Sie kube-state-metrics, Promethus und Alertmanager installieren und konfigurieren.

Installieren Sie kube-state-metrics

Sie können kube-state-metrics mit Helm installieren.

Schritte

1. Fügen Sie das kube-state-metrics Helm-Chart hinzu. Beispiel:

```
helm repo add prometheus-community https://prometheus-
community.github.io/helm-charts
helm repo update
```

2. Wenden Sie die Prometheus ServiceMonitor CRD auf den Cluster an:

```
kubectl apply -f https://raw.githubusercontent.com/prometheus-
operator/prometheus-operator/main/example/prometheus-operator-
crd/monitoring.coreos.com_servicemonitors.yaml
```

3. Erstellen Sie eine Konfigurationsdatei für das Helm-Chart (zum Beispiel `metrics-config.yaml`). Sie können die folgende Beispielkonfiguration an Ihre Umgebung anpassen:

metrics-config.yaml: kube-state-metrics Helm-Chart-Konfiguration

```
---
extraArgs:
  # Collect only custom metrics
  - --custom-resource-state-only=true

customResourceState:
  enabled: true
  config:
    kind: CustomResourceStateMetrics
    spec:
      resources:
      - groupVersionKind:
          group: protect.trident.netapp.io
          kind: "Backup"
          version: "v1"
        labelsFromPath:
          backup_uid: [metadata, uid]
          backup_name: [metadata, name]
          creation_time: [metadata, creationTimestamp]
        metrics:
        - name: backup_info
          help: "Exposes details about the Backup state"
          each:
            type: Info
            info:
              labelsFromPath:
                appVaultReference: ["spec", "appVaultRef"]
                appReference: ["spec", "applicationRef"]
rbac:
  extraRules:
  - apiGroups: ["protect.trident.netapp.io"]
    resources: ["backups"]
    verbs: ["list", "watch"]

# Collect metrics from all namespaces
namespaces: ""

# Ensure that the metrics are collected by Prometheus
prometheus:
  monitor:
    enabled: true
```

4. Installieren Sie kube-state-metrics, indem Sie das Helm-Chart bereitstellen. Beispiel:

```
helm install custom-resource -f metrics-config.yaml prometheus-
community/kube-state-metrics --version 5.21.0
```

5. Konfigurieren Sie kube-state-metrics, um Metriken für die von Trident Protect verwendeten benutzerdefinierten Ressourcen zu generieren, indem Sie den Anweisungen in der "kube-state-metrics Custom Resource Dokumentation" folgen.

Prometheus installieren

Sie können Prometheus installieren, indem Sie den Anweisungen in der "Prometheus-Dokumentation" folgen.

Installieren Sie Alertmanager

Sie können Alertmanager installieren, indem Sie den Anweisungen in der "Alertmanager-Dokumentation" folgen.

Schritt 2: Konfigurieren Sie die Überwachungstools so, dass sie zusammenarbeiten

Nach der Installation der Überwachungstools müssen Sie diese so konfigurieren, dass sie zusammenarbeiten.

Schritte

1. Integrieren Sie kube-state-metrics in Prometheus. Bearbeiten Sie die Prometheus-Konfigurationsdatei (`prometheus.yaml` und fügen Sie die Informationen zum kube-state-metrics-Service hinzu. Beispiel:

   prometheus.yaml: Integration des kube-state-metrics service mit Prometheus

```
---
apiVersion: v1
kind: ConfigMap
metadata:
  name: prometheus-config
  namespace: trident-protect
data:
  prometheus.yaml: |
    global:
      scrape_interval: 15s
    scrape_configs:
      - job_name: 'kube-state-metrics'
        static_configs:
          - targets: ['kube-state-metrics.trident-protect.svc:8080']
```

2. Konfigurieren Sie Prometheus so, dass Warnmeldungen an Alertmanager weitergeleitet werden. Bearbeiten Sie die Prometheus Konfigurationsdatei (`prometheus.yaml` und fügen Sie den folgenden Abschnitt hinzu:

prometheus.yaml: Senden Sie Warnmeldungen an Alertmanager

```
alerting:
  alertmanagers:
    - static_configs:
        - targets:
            - alertmanager.trident-protect.svc:9093
```

Ergebnis

Prometheus kann nun Metriken von kube-state-metrics erfassen und Warnmeldungen an Alertmanager senden. Sie sind jetzt bereit zu konfigurieren, welche Bedingungen eine Warnmeldung auslösen und wohin die Warnmeldungen gesendet werden sollen.

Schritt 3: Benachrichtigungen und Benachrichtigungsziele konfigurieren

Nachdem Sie die Tools so konfiguriert haben, dass sie zusammenarbeiten, müssen Sie konfigurieren, welche Art von Informationen Warnmeldungen auslösen und wohin die Warnmeldungen gesendet werden sollen.

Warnungsbeispiel: Backup-Fehler

Das folgende Beispiel definiert eine kritische Warnung, die ausgelöst wird, wenn der Status der benutzerdefinierten Backup-Ressource auf `Error` für 5 Sekunden oder länger gesetzt ist. Sie können dieses Beispiel an Ihre Umgebung anpassen und diesen YAML-Ausschnitt in Ihre `prometheus.yaml` Konfigurationsdatei einfügen:

rules.yaml: Definiere eine Prometheus-Alarmierung für fehlgeschlagene Backups

```
rules.yaml: |
  groups:
    - name: fail-backup
        rules:
          - alert: BackupFailed
            expr: kube_customresource_backup_info{status="Error"}
            for: 5s
            labels:
              severity: critical
            annotations:
              summary: "Backup failed"
              description: "A backup has failed."
```

Konfigurieren Sie Alertmanager, um Benachrichtigungen an andere Kanäle zu senden

Sie können Alertmanager so konfigurieren, dass Benachrichtigungen an andere Kanäle wie E-Mail, PagerDuty, Microsoft Teams oder andere Benachrichtigungsdienste gesendet werden, indem Sie die jeweilige Konfiguration in der `alertmanager.yaml` Datei angeben.

Das folgende Beispiel konfiguriert Alertmanager so, dass Benachrichtigungen an einen Slack-Kanal gesendet werden. Um dieses Beispiel an Ihre Umgebung anzupassen, ersetzen Sie den Wert des `api_url` Schlüssels durch die in Ihrer Umgebung verwendete Slack-Webhook-URL:

alertmanager.yaml: Senden Sie Warnmeldungen an einen Slack-Kanal

```
data:
  alertmanager.yaml: |
    global:
      resolve_timeout: 5m
    route:
      receiver: 'slack-notifications'
    receivers:
      - name: 'slack-notifications'
        slack_configs:
          - api_url: '<your-slack-webhook-url>'
            channel: '#failed-backups-channel'
            send_resolved: false
```

Generieren Sie ein Trident Protect Support-Bundle

Trident Protect ermöglicht Administratoren, Bundles zu erstellen, die Informationen enthalten, die für den NetApp Support nützlich sind, einschließlich Protokollen, Metriken und Topologieinformationen über die verwalteten Cluster und Apps. Wenn Sie mit dem Internet verbunden sind, können Sie Support-Bundles mithilfe einer benutzerdefinierten Ressourcendatei (CR) auf die NetApp Support Site (NSS) hochladen.

Erstellen Sie ein Support-Bundle mithilfe eines CR

Schritte

1. Erstellen Sie die benutzerdefinierte Ressourcendatei (CR) und benennen Sie sie (zum Beispiel `trident-protect-support-bundle.yaml`).
2. Konfigurieren Sie die folgenden Attribute:
   - **metadata.name**: (*Erforderlich*) Der Name dieser benutzerdefinierten Ressource; wählen Sie einen eindeutigen und sinnvollen Namen für Ihre Umgebung.
   - **spec.triggerType**: (*Erforderlich*) Legt fest, ob das Support-Bundle sofort generiert oder geplant wird. Die geplante Bundle-Generierung erfolgt um 12:00 Uhr UTC. Mögliche Werte:
     - Geplant
     - Handbuch
   - **spec.uploadEnabled**: (*Optional*) Steuert, ob das Support-Bundle nach seiner Generierung auf die NetApp Support-Website hochgeladen werden soll. Wenn nicht angegeben, ist der Standardwert `false`. Mögliche Werte:
     - true
     - false (Standardeinstellung)
   - **spec.dataWindowStart**: (*Optional*) Eine Datumszeichenkette im RFC-3339-Format, die das Datum und die Uhrzeit angibt, zu der das Fenster der im Support-Bundle enthaltenen Daten beginnen soll. Wenn nicht angegeben, wird standardmäßig 24 Stunden zurück angenommen. Das früheste Fensterdatum, das Sie angeben können, liegt 7 Tage zurück.

     Beispiel YAML:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: AutoSupportBundle
metadata:
  name: trident-protect-support-bundle
spec:
  triggerType: Manual
  uploadEnabled: true
  dataWindowStart: 2024-05-05T12:30:00Z
```

3. Nachdem Sie die `trident-protect-support-bundle.yaml` Datei mit den korrekten Werten gefüllt haben, wenden Sie die CR an:

```
kubectl apply -f trident-protect-support-bundle.yaml -n trident-
protect
```

Erstellen Sie ein Support-Bundle mithilfe der CLI

Schritte

1. Erstellen Sie das Support-Bundle, indem Sie die Werte in Klammern durch Informationen aus Ihrer

Umgebung ersetzen. Das `trigger-type` bestimmt, ob das Bundle sofort erstellt wird oder ob die Erstellungszeit durch den Zeitplan vorgegeben ist, und kann `Manual` oder `Scheduled` sein. Die Standardeinstellung ist `Manual`.

Beispiel:

```
tridentctl-protect create autosupportbundle <my-bundle-name>
--trigger-type <trigger-type> -n trident-protect
```

Überwachen und rufen Sie das Support-Paket ab

Nachdem Sie mit einer der beiden Methoden ein Support-Bundle erstellt haben, können Sie den Generierungsfortschritt überwachen und es auf Ihr lokales System abrufen.

Schritte

1. Warten Sie, bis der `status.generationState` den `Completed` Status erreicht. Sie können den Generierungsfortschritt mit dem folgenden Befehl überwachen:

   ```
   kubectl get autosupportbundle trident-protect-support-bundle -n trident-
   protect
   ```

2. Rufen Sie das Support-Bundle auf Ihr lokales System ab. Rufen Sie den Kopierbefehl aus dem abgeschlossenen AutoSupport-Bundle ab:

   ```
   kubectl describe autosupportbundle trident-protect-support-bundle -n
   trident-protect
   ```

   Suchen Sie den `kubectl cp` Befehl in der Ausgabe und führen Sie ihn aus, wobei Sie das Argument Ziel durch Ihr bevorzugtes lokales Verzeichnis ersetzen.

Trident Protect aktualisieren

Sie können Trident Protect auf die neueste Version aktualisieren, um neue Funktionen oder Fehlerbehebungen zu nutzen.

HINWEIS

- Beim Upgrade von Version 24.10 kann es vorkommen, dass während des Upgrades laufende Snapshots fehlschlagen. Dieses Fehlschlagen verhindert nicht, dass zukünftige Snapshots, egal ob manuell oder geplant, erstellt werden können. Wenn ein Snapshot während des Upgrades fehlschlägt, können Sie manuell einen neuen Snapshot erstellen, um sicherzustellen, dass Ihre Anwendung geschützt ist.

  Um mögliche Fehler zu vermeiden, können Sie vor dem Upgrade alle Snapshot-Zeitpläne deaktivieren und sie danach wieder aktivieren. Dies führt jedoch dazu, dass während des Upgrade-Zeitraums geplante Snapshots fehlen.

- Bei Installationen in privaten Registrys stellen Sie sicher, dass das erforderliche Helm-Chart und die Images für die Zielversion in Ihrer privaten Registry verfügbar sind, und überprüfen Sie, ob Ihre benutzerdefinierten Helm-Werte mit der neuen Chart-Version kompatibel sind. Weitere Informationen finden sich unter "Installieren Sie Trident Protect aus einem privaten Registry.".

Schritt 1: Wählen Sie eine Version

Trident Protect-Versionen folgen einer datumsbasierten `YY.MM` Namenskonvention, wobei „YY“ die letzten beiden Ziffern des Jahres und „MM“ der Monat sind. Dot-Releases folgen einer `YY.MM.X` Konvention, wobei „X“ das Patch-Level ist. Sie wählen die Version aus, auf die Sie aktualisieren möchten, basierend auf der Version, von der Sie aktualisieren.

- Sie können ein direktes Upgrade auf jede Zielversion durchführen, die innerhalb eines Vier-Versions-Fensters Ihrer installierten Version liegt. Beispielsweise können Sie direkt von 24.10 (oder jeder 24.10 dot release) auf 25.10 aktualisieren.
- Wenn Sie von einer Version außerhalb des Vier-Versions-Fensters aktualisieren, ist ein mehrstufiges Upgrade erforderlich. Die Upgrade-Anweisungen für die "frühere Version" Version, von der Sie aktualisieren, sind zu verwenden, um auf die neueste Version innerhalb des Vier-Versions-Fensters zu aktualisieren. Beispielsweise, wenn Version 24.10 verwendet wird und ein Upgrade auf 26.02 erfolgen soll:
  a. Erstes Upgrade von 24.10 auf 25.02.
  b. Führen Sie dann ein Upgrade von 25.02 auf 26.02 durch.

Schritt 2: Trident Protect aktualisieren

Um Trident Protect zu aktualisieren, führen Sie die folgenden Schritte aus.

Schritte

1. Aktualisieren Sie das Trident Helm-Repository:

```
helm repo update
```

2. Aktualisieren Sie die Trident Protect CRDs:

HINWEIS Dieser Schritt ist erforderlich, wenn Sie von einer Version vor 25.06 aktualisieren, da die CRDs jetzt im Trident Protect Helm chart enthalten sind.

   a. Führen Sie diesen Befehl aus, um die Verwaltung von CRDs von `trident-protect-crds` zu `trident-protect` zu verschieben:

```
kubectl get crd | grep protect.trident.netapp.io | awk '{print $1}' |
xargs -I {} kubectl patch crd {} --type merge -p '{"metadata":
{"annotations":{"meta.helm.sh/release-name": "trident-protect"}}}'
```

b. Führen Sie diesen Befehl aus, um das Helm-Secret für das `trident-protect-crds` Chart zu löschen:

HINWEIS Deinstallieren Sie das `trident-protect-crds` Chart nicht mit Helm, da dies Ihre CRDs und alle zugehörigen Daten entfernen könnte.

```
kubectl delete secret -n trident-protect -l name=trident-protect-
crds,owner=helm
```

3. Trident Protect aktualisieren:

```
helm upgrade trident-protect netapp-trident-protect/trident-protect
--version 100.2602.0 --namespace trident-protect
```

HINWEIS Sie können den Protokollierungsgrad während des Upgrades konfigurieren, indem Sie `--set logLevel=debug` zum Upgrade-Befehl hinzufügen. Der Standard-Protokollierungsgrad ist `warn`. Debug-Protokollierung wird für die Fehlerbehebung empfohlen, da sie NetApp Support hilft, Probleme zu diagnostizieren, ohne dass Änderungen am Protokollierungsgrad oder eine Problemreproduktion erforderlich sind.

Anwendungen verwalten und schützen

Verwenden Sie Trident Protect AppVault-Objekte, um Buckets zu verwalten

Die Bucket-Custom-Resource (CR) für Trident Protect ist als AppVault bekannt. AppVault-Objekte sind die deklarative Kubernetes-Workflow-Darstellung eines Speicher-Buckets. Ein AppVault-CR enthält die Konfigurationen, die erforderlich sind, damit ein Bucket in Schutzvorgängen wie Backups, Snapshots, Wiederherstellungsvorgängen und SnapMirror-Replikation verwendet werden kann. Nur Administratoren können AppVaults erstellen.

Sie müssen ein AppVault CR manuell oder über die Befehlszeile erstellen, wenn Sie Datensicherungsoperationen an einer Anwendung durchführen. Das AppVault CR ist spezifisch für Ihre Umgebung, und Sie können die Beispiele auf dieser Seite als Leitfaden verwenden, wenn Sie AppVault CRs erstellen.

HINWEIS Stellen Sie sicher, dass die AppVault CR auf dem Cluster vorhanden ist, auf dem Trident Protect installiert ist. Wenn die AppVault CR nicht existiert oder Sie nicht darauf zugreifen können, zeigt die Befehlszeile einen Fehler an.

Konfigurieren Sie die AppVault-Authentifizierung und Passwörter

Bevor Sie ein AppVault CR erstellen, stellen Sie sicher, dass der AppVault und der von Ihnen gewählte Datenmover sich beim Anbieter und allen zugehörigen Ressourcen authentifizieren können.

Passwörter für das Data mover-Repository

Wenn Sie AppVault-Objekte mit CRs oder dem Trident Protect CLI-Plugin erstellen, können Sie ein Kubernetes-Secret mit benutzerdefinierten Passwörtern für die Restic- und Kopia-Verschlüsselung angeben. Wenn Sie kein Secret angeben, verwendet Trident Protect ein Standardpasswort.

- Wenn Sie AppVault-CRs manuell erstellen, verwenden Sie das Feld **spec.dataMoverPasswordSecretRef**, um das Secret anzugeben.
- Beim Erstellen von AppVault-Objekten mit der Trident Protect CLI verwenden Sie das `--data-mover -password-secret-ref` Argument, um das Geheimnis anzugeben.

Erstellen Sie ein Passwortgeheimnis für das Data Mover Repository

Verwenden Sie die folgenden Beispiele, um das Passwortgeheimnis zu erstellen. Wenn Sie AppVault-Objekte erstellen, können Sie Trident Protect anweisen, dieses Geheimnis zur Authentifizierung beim Datenübertragungs-Repository zu verwenden.

HINWEIS

- Je nachdem, welchen Datenmover Sie verwenden, müssen Sie nur das entsprechende Passwort für diesen Datenmover angeben. Wenn Sie beispielsweise Restic verwenden und nicht planen, Kopia in Zukunft zu nutzen, können Sie beim Erstellen des Geheimnisses nur das Restic-Passwort angeben.
- Bewahren Sie das Passwort sicher auf. Sie benötigen es, um Daten auf demselben Cluster oder auf einem anderen Cluster wiederherzustellen. Wenn der Cluster oder das `trident-protect` Namespace gelöscht wird, können Sie Ihre Backups oder Snapshots ohne das Passwort nicht wiederherstellen.

Verwenden Sie einen CR

```
---
apiVersion: v1
data:
  KOPIA_PASSWORD: <base64-encoded-password>
  RESTIC_PASSWORD: <base64-encoded-password>
kind: Secret
metadata:
  name: my-optional-data-mover-secret
  namespace: trident-protect
type: Opaque
```

Verwenden Sie die Befehlszeile

```
kubectl create secret generic my-optional-data-mover-secret \
--from-literal=KOPIA_PASSWORD=<plain-text-password> \
--from-literal=RESTIC_PASSWORD=<plain-text-password> \
-n trident-protect
```

S3-kompatible Speicher-IAM-Berechtigungen

Wenn Sie auf S3-kompatiblen Speicher wie Amazon S3, Generic S3, "StorageGrid S3", oder "ONTAP S3" mit Trident Protect zugreifen, müssen Sie sicherstellen, dass die von Ihnen bereitgestellten Benutzeranmeldeinformationen über die erforderlichen Berechtigungen für den Zugriff auf den Bucket verfügen. Im Folgenden finden Sie ein Beispiel für eine Richtlinie, die die minimal erforderlichen Berechtigungen für den Zugriff mit Trident Protect gewährt. Sie können diese Richtlinie auf den Benutzer anwenden, der die S3-kompatiblen Bucket-Richtlinien verwaltet.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:GetObject",
        "s3:ListBucket",
        "s3:DeleteObject"
      ],
      "Resource": "*"
    }
  ]
}
```

Weitere Informationen zu Amazon S3-Richtlinien finden Sie in den Beispielen im "Amazon S3-Dokumentation".

EKS Pod Identity für Amazon S3 (AWS) Authentifizierung

Trident Protect unterstützt EKS Pod Identity für Kopia data mover-Operationen. Diese Funktion ermöglicht sicheren Zugriff auf S3-Buckets, ohne AWS-Anmeldeinformationen in Kubernetes-Secrets speichern zu müssen.

Anforderungen für EKS Pod Identity mit Trident Protect

Bevor Sie EKS Pod Identity mit Trident Protect verwenden, stellen Sie Folgendes sicher:

- In Ihrem EKS-Cluster ist Pod Identity aktiviert.
- Sie haben eine IAM-Rolle mit den erforderlichen S3-Bucket-Berechtigungen erstellt. Weitere Informationen finden Sie unter "S3-kompatible Speicher-IAM-Berechtigungen".
- Die IAM-Rolle ist den folgenden Trident Protect service accounts zugeordnet:
  - `<trident-protect>-controller-manager`
  - `<trident-protect>-resource-backup`
  - `<trident-protect>-resource-restore`
  - `<trident-protect>-resource-delete`

Detaillierte Anweisungen zum Aktivieren von Pod Identity und zum Zuordnen von IAM-Rollen zu Servicekonten finden Sie in der "AWS EKS Pod Identity-Dokumentation".

AppVault-Konfiguration Bei Verwendung von EKS Pod Identity konfigurieren Sie Ihre AppVault-CR mit dem `useIAM: true`-Flag anstelle expliziter Anmeldeinformationen:

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: eks-protect-vault
  namespace: trident-protect
spec:
  providerType: AWS
  providerConfig:
    s3:
      bucketName: trident-protect-aws
      endpoint: s3.example.com
      useIAM: true
```

AppVault Schlüsselgenerierungsbeispiele für Cloud-Anbieter

Beim Definieren eines AppVault CR müssen Sie Anmeldeinformationen angeben, um auf die vom Anbieter gehosteten Ressourcen zuzugreifen, es sei denn, Sie verwenden IAM-Authentifizierung. Wie Sie die Schlüssel für die Anmeldeinformationen generieren, hängt vom jeweiligen Anbieter ab. Im Folgenden finden Sie Beispiele für die Schlüsselgenerierung über die Befehlszeile für verschiedene Anbieter. Sie können die folgenden Beispiele verwenden, um Schlüssel für die Anmeldeinformationen jedes Cloud-Anbieters zu erstellen.

Google Cloud

```
kubectl create secret generic <secret-name> \
--from-file=credentials=<mycreds-file.json> \
-n trident-protect
```

Amazon S3 (AWS)

```
kubectl create secret generic <secret-name> \
--from-literal=accessKeyID=<objectstorage-accesskey> \
--from-literal=secretAccessKey=<amazon-s3-trident-protect-src-bucket
-secret> \
-n trident-protect
```

Microsoft Azure

```
kubectl create secret generic <secret-name> \
--from-literal=accountKey=<secret-name> \
-n trident-protect
```

Generisches S3

```
kubectl create secret generic <secret-name> \
--from-literal=accessKeyID=<objectstorage-accesskey> \
--from-literal=secretAccessKey=<generic-s3-trident-protect-src-bucket
-secret> \
-n trident-protect
```

ONTAP S3

```
kubectl create secret generic <secret-name> \
--from-literal=accessKeyID=<objectstorage-accesskey> \
--from-literal=secretAccessKey=<ontap-s3-trident-protect-src-bucket
-secret> \
-n trident-protect
```

StorageGrid S3

```
kubectl create secret generic <secret-name> \
--from-literal=accessKeyID=<objectstorage-accesskey> \
--from-literal=secretAccessKey=<storagegrid-s3-trident-protect-src
-bucket-secret> \
-n trident-protect
```

AppVault-Erstellungsbeispiele

Die folgenden sind Beispiel-AppVault-Definitionen für jeden Anbieter.

AppVault CR-Beispiele

Anhand der folgenden CR-Beispiele können Sie AppVault-Objekte für jeden Cloud-Anbieter erstellen.

HINWEIS

- Optional können Sie ein Kubernetes-Secret angeben, das benutzerdefinierte Passwörter für die Verschlüsselung der Restic- und Kopia-Repositories enthält. Weitere Informationen sind unter Passwörter für das Data mover-Repository verfügbar.
- Für Amazon S3 (AWS) AppVault-Objekte können Sie optional ein sessionToken angeben, was nützlich ist, wenn Sie Single Sign-On (SSO) zur Authentifizierung verwenden. Dieses Token wird erstellt, wenn Sie Schlüssel für den Anbieter in AppVault Schlüsselgenerierungsbeispiele für Cloud-Anbieter generieren.
- Für S3 AppVault-Objekte können Sie optional eine Egress-Proxy-URL für ausgehenden S3-Datenverkehr mithilfe des `spec.providerConfig.S3.proxyURL` Schlüssels angeben.

Google Cloud

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: gcp-trident-protect-src-bucket
  namespace: trident-protect
spec:
  dataMoverPasswordSecretRef: my-optional-data-mover-secret
  providerType: GCP
  providerConfig:
    gcp:
      bucketName: trident-protect-src-bucket
      projectID: project-id
  providerCredentials:
    credentials:
      valueFromSecret:
        key: credentials
        name: gcp-trident-protect-src-bucket-secret
```

Amazon S3 (AWS)

```
---
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: amazon-s3-trident-protect-src-bucket
  namespace: trident-protect
spec:
  dataMoverPasswordSecretRef: my-optional-data-mover-secret
  providerType: AWS
  providerConfig:
    s3:
      bucketName: trident-protect-src-bucket
      endpoint: s3.example.com
      proxyURL: http://10.1.1.1:3128
  providerCredentials:
    accessKeyID:
      valueFromSecret:
        key: accessKeyID
        name: s3-secret
    secretAccessKey:
      valueFromSecret:
        key: secretAccessKey
        name: s3-secret
    sessionToken:
      valueFromSecret:
        key: sessionToken
        name: s3-secret
```

HINWEIS Für EKS-Umgebungen, die Pod Identity mit Kopia Data Mover verwenden, können Sie den `providerCredentials` Abschnitt entfernen und `useIAM: true` unter der `s3` Konfiguration hinzufügen.

Microsoft Azure

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: azure-trident-protect-src-bucket
  namespace: trident-protect
spec:
  dataMoverPasswordSecretRef: my-optional-data-mover-secret
  providerType: Azure
  providerConfig:
    azure:
      accountName: account-name
      bucketName: trident-protect-src-bucket
  providerCredentials:
    accountKey:
      valueFromSecret:
        key: accountKey
        name: azure-trident-protect-src-bucket-secret
```

Generisches S3

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: generic-s3-trident-protect-src-bucket
  namespace: trident-protect
spec:
  dataMoverPasswordSecretRef: my-optional-data-mover-secret
  providerType: GenericS3
  providerConfig:
    s3:
      bucketName: trident-protect-src-bucket
      endpoint: s3.example.com
      proxyURL: http://10.1.1.1:3128
  providerCredentials:
    accessKeyID:
      valueFromSecret:
        key: accessKeyID
        name: s3-secret
    secretAccessKey:
      valueFromSecret:
        key: secretAccessKey
        name: s3-secret
```

ONTAP S3

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: ontap-s3-trident-protect-src-bucket
  namespace: trident-protect
spec:
  dataMoverPasswordSecretRef: my-optional-data-mover-secret
  providerType: OntapS3
  providerConfig:
    s3:
      bucketName: trident-protect-src-bucket
      endpoint: s3.example.com
      proxyURL: http://10.1.1.1:3128
  providerCredentials:
    accessKeyID:
      valueFromSecret:
        key: accessKeyID
        name: s3-secret
    secretAccessKey:
      valueFromSecret:
        key: secretAccessKey
        name: s3-secret
```

StorageGrid S3

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: storagegrid-s3-trident-protect-src-bucket
  namespace: trident-protect
spec:
  dataMoverPasswordSecretRef: my-optional-data-mover-secret
  providerType: StorageGridS3
  providerConfig:
    s3:
      bucketName: trident-protect-src-bucket
      endpoint: s3.example.com
      proxyURL: http://10.1.1.1:3128
  providerCredentials:
    accessKeyID:
      valueFromSecret:
        key: accessKeyID
        name: s3-secret
    secretAccessKey:
      valueFromSecret:
        key: secretAccessKey
        name: s3-secret
```

AppVault Erstellungsbeispiele mit der Trident Protect CLI

Sie können die folgenden CLI-Befehlsbeispiele verwenden, um AppVault CRs für jeden Anbieter zu erstellen.

HINWEIS

- Optional können Sie ein Kubernetes-Secret angeben, das benutzerdefinierte Passwörter für die Verschlüsselung der Restic- und Kopia-Repositories enthält. Weitere Informationen sind unter Passwörter für das Data mover-Repository verfügbar.
- Für S3-AppVault-Objekte können Sie optional eine Egress-Proxy-URL für ausgehenden S3-Datenverkehr mithilfe des `--proxy-url <ip_address:port>` Arguments angeben.

Google Cloud

```
tridentctl-protect create vault GCP <vault-name> \
--bucket <mybucket> \
--project <my-gcp-project> \
--secret <secret-name>/credentials \
--data-mover-password-secret-ref <my-optional-data-mover-secret> \
-n trident-protect
```

Amazon S3 (AWS)

```
tridentctl-protect create vault AWS <vault-name> \
--bucket <bucket-name> \
--secret  <secret-name>  \
--endpoint <s3-endpoint> \
--data-mover-password-secret-ref <my-optional-data-mover-secret> \
-n trident-protect
```

Microsoft Azure

```
tridentctl-protect create vault Azure <vault-name> \
--account <account-name> \
--bucket <bucket-name> \
--secret <secret-name> \
--data-mover-password-secret-ref <my-optional-data-mover-secret> \
-n trident-protect
```

Generisches S3

```
tridentctl-protect create vault GenericS3 <vault-name> \
--bucket <bucket-name> \
--secret  <secret-name>  \
--endpoint <s3-endpoint> \
--data-mover-password-secret-ref <my-optional-data-mover-secret> \
-n trident-protect
```

ONTAP S3

```
tridentctl-protect create vault OntapS3 <vault-name> \
--bucket <bucket-name> \
--secret  <secret-name>  \
--endpoint <s3-endpoint> \
--data-mover-password-secret-ref <my-optional-data-mover-secret> \
-n trident-protect
```

StorageGrid S3

```
tridentctl-protect create vault StorageGridS3 <vault-name> \
--bucket <bucket-name> \
--secret  <secret-name>  \
--endpoint <s3-endpoint> \
--data-mover-password-secret-ref <my-optional-data-mover-secret> \
-n trident-protect
```

Unterstützte `providerConfig.s3` Konfigurationsoptionen

Siehe die folgende Tabelle für die S3-Provider-Konfigurationsoptionen:

Parameter	Beschreibung	Standard	Beispiel
`providerConfig.s3.skipCertValidation`	SSL/TLS-Zertifikatsüberprüfung deaktivieren.	false	"true", "false"
`providerConfig.s3.secure`	Aktivieren Sie die sichere HTTPS-Kommunikation mit dem S3-Endpunkt.	true	"true", "false"
`providerConfig.s3.proxyURL`	Geben Sie die URL des Proxyservers an, der zur Verbindung mit S3 verwendet wird.	Keine	http://proxy.example.com:8080
`providerConfig.s3.rootCA`	Stellen Sie ein benutzerdefiniertes Root-CA-Zertifikat für die SSL/TLS-Verifizierung bereit.	Keine	"CN=MyCustomCA"
`providerConfig.s3.useIAM`	Aktivieren Sie die IAM-Authentifizierung für den Zugriff auf S3-Buckets. Gilt für EKS Pod Identity.	false	wahr, falsch

Informationen zu AppVault anzeigen

Sie können das Trident Protect CLI-Plugin verwenden, um Informationen über AppVault-Objekte anzuzeigen, die Sie auf dem Cluster erstellt haben.

Schritte

1. Den Inhalt eines AppVault Objekts anzeigen:

```
tridentctl-protect get appvaultcontent gcp-vault \
--show-resources all \
-n trident-protect
```

Beispielausgabe:

```
+-------------+-------+----------+-----------------------------
+---------------------------+
|   CLUSTER   |  APP  |   TYPE   |            NAME             |
TIMESTAMP         |
+-------------+-------+----------+-----------------------------
+---------------------------+
|             | mysql | snapshot | mysnap                      | 2024-
08-09 21:02:11 (UTC) |
| production1 | mysql | snapshot | hourly-e7db6-20240815180300 | 2024-
08-15 18:03:06 (UTC) |
| production1 | mysql | snapshot | hourly-e7db6-20240815190300 | 2024-
08-15 19:03:06 (UTC) |
| production1 | mysql | snapshot | hourly-e7db6-20240815200300 | 2024-
08-15 20:03:06 (UTC) |
| production1 | mysql | backup   | hourly-e7db6-20240815180300 | 2024-
08-15 18:04:25 (UTC) |
| production1 | mysql | backup   | hourly-e7db6-20240815190300 | 2024-
08-15 19:03:30 (UTC) |
| production1 | mysql | backup   | hourly-e7db6-20240815200300 | 2024-
08-15 20:04:21 (UTC) |
| production1 | mysql | backup   | mybackup5                   | 2024-
08-09 22:25:13 (UTC) |
|             | mysql | backup   | mybackup                    | 2024-
08-09 21:02:52 (UTC) |
+-------------+-------+----------+-----------------------------
+---------------------------+
```

2. Optional können Sie den AppVaultPath für jede Ressource anzeigen, indem Sie das Flag `--show-paths` verwenden.

   Der Clustername in der ersten Spalte der Tabelle ist nur verfügbar, wenn bei der Trident Protect helm-Installation ein Clustername angegeben wurde. Zum Beispiel: `--set clusterName=production1`.

Entfernen Sie ein AppVault

Sie können ein AppVault-Objekt jederzeit entfernen.

HINWEIS	Entfernen Sie den `finalizers` Schlüssel im AppVault CR nicht, bevor Sie das AppVault Objekt löschen. Wenn Sie dies tun, kann dies zu Restdaten im AppVault Bucket und zu verwaisten Ressourcen im Cluster führen.

Bevor Sie beginnen

Stellen Sie sicher, dass Sie alle Snapshot- und Backup-CRs gelöscht haben, die von dem AppVault verwendet werden, das Sie löschen möchten.

Entfernen Sie ein AppVault mithilfe der Kubernetes-CLI

1. Entfernen Sie das AppVault-Objekt, wobei Sie `appvault-name` durch den Namen des zu entfernenden AppVault-Objekts ersetzen:

```
kubectl delete appvault <appvault-name> \
-n trident-protect
```

Entfernen Sie ein AppVault mit der Trident Protect CLI

1. Entfernen Sie das AppVault-Objekt, wobei Sie `appvault-name` durch den Namen des zu entfernenden AppVault-Objekts ersetzen:

```
tridentctl-protect delete appvault <appvault-name> \
-n trident-protect
```

Definieren Sie eine Anwendung für das Management mit Trident Protect

Sie können eine Anwendung, die Sie mit Trident Protect verwalten möchten, definieren, indem Sie eine Anwendungs-CR und eine zugehörige AppVault CR erstellen.

Erstellen Sie eine AppVault CR

Sie müssen ein AppVault-CR erstellen, das bei der Durchführung von Datensicherungsoperationen an der Anwendung verwendet wird, und das AppVault-CR muss sich auf dem Cluster befinden, auf dem Trident Protect installiert ist. Das AppVault-CR ist spezifisch für Ihre Umgebung; Beispiele für AppVault-CRs finden Sie unter "AppVault benutzerdefinierte Ressourcen."

Eine Anwendung definieren

Sie müssen jede Anwendung, die Sie mit Trident Protect verwalten möchten, definieren. Sie können eine Anwendung zur Verwaltung entweder manuell durch Erstellen eines Anwendungs-CR oder mithilfe der Trident Protect CLI definieren.

Fügen Sie eine Anwendung mithilfe eines CR hinzu

Schritte

1. Erstellen Sie die CR-Datei der Zielanwendung:
   a. Erstellen Sie die benutzerdefinierte Ressourcendatei (CR) und benennen Sie sie (zum Beispiel `maria-app.yaml`).
   b. Konfigurieren Sie die folgenden Attribute:
      - **metadata.name**: (*Erforderlich*) Der Name der benutzerdefinierten Anwendungsressource. Merken Sie sich den Namen, den Sie wählen, da andere für Schutzvorgänge benötigte CR-Dateien auf diesen Wert verweisen.
      - **spec.includedNamespaces**: (*Erforderlich*) Verwenden Sie Namespace und Label-Selektor, um die Namespaces und Ressourcen anzugeben, die die Anwendung verwendet. Der Anwendungsnamespace muss Teil dieser Liste sein. Der Label-Selektor ist optional und kann verwendet werden, um Ressourcen innerhalb jedes angegebenen Namespace zu filtern.
      - **spec.includedClusterScopedResources**: (*Optional*) Verwenden Sie dieses Attribut, um Cluster-Scoped-Ressourcen anzugeben, die in die Anwendungsdefinition aufgenommen werden sollen. Mit diesem Attribut können Sie diese Ressourcen anhand ihrer Gruppe, Version, Art und Bezeichnungen auswählen.
        - **groupVersionKind**: (*Erforderlich*) Gibt die API-Gruppe, die Version und die Art der clusterweiten Ressource an.
        - **labelSelector**: (*Optional*) Filtert die clusterweiten Ressourcen anhand ihrer Labels.
      - **metadata.annotations.protect.trident.netapp.io/skip-vm-freeze**: (*Optional*) Diese Annotation ist nur für Anwendungen relevant, die von virtuellen Maschinen aus definiert werden, z. B. in KubeVirt-Umgebungen, in denen das Dateisystem vor Snapshots eingefroren wird. Legen Sie fest, ob diese Anwendung während eines Snapshots auf das Dateisystem schreiben darf. Ist die Option auf true gesetzt, ignoriert die Anwendung die globale Einstellung und kann während eines Snapshots auf das Dateisystem schreiben. Ist die Option auf false gesetzt, ignoriert die Anwendung die globale Einstellung und das Dateisystem wird während eines Snapshots eingefroren. Wird die Option angegeben, die Anwendung aber keine virtuellen Maschinen in der Anwendungsdefinition hat, wird die Annotation ignoriert. Wird sie nicht angegeben, folgt die Anwendung der "globale Trident Protect freeze-Einstellung".

Falls Sie diese Annotation nachträglich anwenden müssen, nachdem eine Anwendung bereits erstellt wurde, können Sie den folgenden Befehl verwenden:

```
kubectl annotate application -n <application CR namespace> <application CR name> protect.trident.netapp.io/skip-vm-freeze="true"
```

+
Beispiel YAML:

+

```
apiVersion: protect.trident.netapp.io/v1
kind: Application
metadata:
  annotations:
    protect.trident.netapp.io/skip-vm-freeze: "false"
  name: my-app-name
  namespace: my-app-namespace
spec:
  includedNamespaces:
    - namespace: namespace-1
      labelSelector:
        matchLabels:
          app: example-app
    - namespace: namespace-2
      labelSelector:
        matchLabels:
          app: another-example-app
  includedClusterScopedResources:
    - groupVersionKind:
        group: rbac.authorization.k8s.io
        kind: ClusterRole
        version: v1
      labelSelector:
        matchLabels:
          mylabel: test
```

1. (*Optional*) Falls erforderlich, können Sie der gleichen CR eine Ressourcenfilterung hinzufügen, um bestimmte Ressourcen einzuschließen oder auszuschließen:

   - **Beispiel für einen generischen Filter:**

     - **resourceFilter.resourceSelectionCriteria**: (Für die Filterung erforderlich) Verwenden Sie `Include` oder `Exclude`, um eine in resourceMatchers definierte Ressource ein- oder auszuschließen. Fügen Sie die folgenden resourceMatchers-Parameter hinzu, um die Ressourcen zu definieren, die ein- oder auszuschließen sind:

       - **resourceFilter.resourceMatchers**: Ein Array von resourceMatcher-Objekten. Wenn Sie mehrere Elemente in diesem Array definieren, werden diese mit einer ODER-Verknüpfung verglichen und die Felder innerhalb jedes Elements (group, kind, version) werden mit einer UND-Verknüpfung verglichen.
       - **resourceMatchers[].group**: (*Optional*) Gruppe der zu filternden Ressource.

- **resourceMatchers[].kind**: (*Optional*) Art der zu filternden Ressource.
- **resourceMatchers[].version**: (*Optional*) Version der zu filternden Ressource.
- **resourceMatchers[].names**: (*Optional*) Namen im Kubernetes metadata.name-Feld der Ressource, die gefiltert werden soll.
- **resourceMatchers[].namespaces**: (*Optional*) Namespaces im Kubernetes metadata.name-Feld der Ressource, die gefiltert werden soll.
- **resourceMatchers[].labelSelectors**: (*Optional*) Label-Selektorzeichenfolge im Kubernetes-Metadatenfeld name der Ressource, wie definiert in der "Kubernetes-Dokumentation". Zum Beispiel: `"trident.netapp.io/os=linux"`.

HINWEIS	Wenn sowohl `resourceFilter` als auch `labelSelector` verwendet werden, wird `resourceFilter` zuerst ausgeführt und anschließend `labelSelector` auf die resultierenden Ressourcen angewendet.

Beispiel:

```
spec:
  resourceFilter:
    resourceSelectionCriteria: "Include"
    resourceMatchers:
      - group: my-resource-group-1
        kind: my-resource-kind-1
        version: my-resource-version-1
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
      - group: my-resource-group-2
        kind: my-resource-kind-2
        version: my-resource-version-2
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
```

◦ **Beispiel für einen reinen PVC-Filter:**

Um eine reine PVC-Anwendung zu definieren, müssen Sie auch `PersistentVolume` und `VolumeSnapshotClass` im Ressourcenfilter angeben. Snapshot- und Backup-Vorgänge hängen von `PersistentVolume` (dem clusterweiten Volume, das an jedes PVC gebunden ist) und `VolumeSnapshotClass` (dem Snapshot-Treiber) ab und schlagen ohne diese fehl. Beispiel:

```
apiVersion: protect.trident.netapp.io/v1
kind: Application
metadata:
  name: my-pvc-app
  namespace: my-app-namespace
spec:
  includedNamespaces:
  - namespace: my-app-namespace
  resourceFilter:
    resourceMatchers:
    - kind: PersistentVolumeClaim
      version: v1
    - kind: PersistentVolume
      version: v1
    - kind: VolumeSnapshotClass
      version: v1
    resourceSelectionCriteria: Include
```

2. Nachdem Sie die Anwendungs-CR erstellt haben, die zu Ihrer Umgebung passt, wenden Sie die CR an. Beispiel:

```
kubectl apply -f maria-app.yaml
```

Schritte

1. Erstellen und wenden Sie die Anwendungsdefinition anhand eines der folgenden Beispiele an, wobei Sie die Werte in Klammern durch Informationen aus Ihrer Umgebung ersetzen. Sie können Namensräume und Ressourcen in die Anwendungsdefinition einbinden, indem Sie durch Kommas getrennte Listen mit den in den Beispielen gezeigten Argumenten verwenden.

   Beim Erstellen einer App können Sie optional eine Annotation verwenden, um festzulegen, ob die Anwendung während eines Snapshots auf das Dateisystem schreiben darf. Dies gilt nur für Anwendungen, die von virtuellen Maschinen definiert werden, beispielsweise in KubeVirt-Umgebungen, in denen das Dateisystem vor Snapshots eingefroren wird. Wenn Sie die Annotation auf `true` setzen, ignoriert die Anwendung die globale Einstellung und kann während eines Snapshots auf das Dateisystem schreiben. Wenn Sie sie auf `false` setzen, ignoriert die Anwendung die globale Einstellung und das Dateisystem wird während eines Snapshots eingefroren. Wenn Sie die Annotation verwenden, die Anwendung aber keine virtuellen Maschinen in der Anwendungsdefinition hat, wird die Annotation ignoriert. Wenn Sie die Annotation nicht verwenden, folgt die Anwendung der "globale Trident Protect freeze-Einstellung".

   Um die Annotation anzugeben, wenn Sie die CLI zum Erstellen einer Anwendung verwenden, können Sie das `--annotation` Flag verwenden.

   - Erstellen Sie die Anwendung und verwenden Sie die globale Einstellung für das Einfrieren des Dateisystems:

```
tridentctl-protect create application <my_new_app_cr_name>
--namespaces <namespaces_to_include> --csr
<cluster_scoped_resources_to_include> --namespace <my-app-
namespace>
```

- Erstellen Sie die Anwendung und konfigurieren Sie die lokale Anwendungseinstellung für das Dateisystem-Einfrierverhalten:

```
tridentctl-protect create application <my_new_app_cr_name>
--namespaces <namespaces_to_include> --csr
<cluster_scoped_resources_to_include> --namespace <my-app-
namespace> --annotation protect.trident.netapp.io/skip-vm-freeze
=<"true"|"false">
```

- Sie können `--resource-filter-include` und `--resource-filter-exclude` Flags verwenden, um Ressourcen basierend auf `resourceSelectionCriteria` wie Gruppe, Art, Version, Bezeichnungen, Namen und Namensräumen ein- oder auszuschließen, wie im folgenden Beispiel gezeigt:

```
tridentctl-protect create application <my_new_app_cr_name>
--namespaces <namespaces_to_include> --csr
<cluster_scoped_resources_to_include> --namespace <my-app-
namespace> --resource-filter-include
'[{"Group":"apps","Kind":"Deployment","Version":"v1","Names":["my
-deployment"],"Namespaces":["my
-namespace"],"LabelSelectors":["app=my-app"]}]'
```

- Um eine reine PVC-Anwendung zu definieren, müssen Sie auch `PersistentVolume` und `VolumeSnapshotClass` im Ressourcenfilter angeben. Snapshot- und Backup-Vorgänge hängen von `PersistentVolume` (dem clusterweiten Volume, das an jedes PVC gebunden ist) und `VolumeSnapshotClass` (dem Snapshot-Treiber) ab und schlagen ohne diese fehl. Beispiel:

```
tridentctl-protect create app my-pvc-app --namespaces <my-app-
namespace> --resource-filter-include
'[{"Kind":"PersistentVolumeClaim","Version":"v1"},{"Kind":"Persis
tentVolume","Version":"v1"},{"Kind":"VolumeSnapshotClass","Versio
n":"v1"}]' -n <my-app-namespace>
```

Schützen Sie Anwendungen mit Trident Protect

Sie können alle von Trident Protect verwalteten Apps schützen, indem Sie Snapshots und Backups mithilfe einer automatisierten Datensicherungsstrategie oder nach Bedarf

erstellen.

HINWEIS	Sie können Trident Protect so konfigurieren, dass Dateisysteme während Datensicherungsoperationen eingefroren und wieder freigegeben werden. "Erfahren Sie mehr über die Konfiguration des Einfrierens von Dateisystemen mit Trident Protect".

Erstellen Sie einen On-Demand-Snapshot

Sie können jederzeit einen On-Demand-Snapshot erstellen.

HINWEIS	Clusterbezogene Ressourcen werden in eine Sicherung, einen Snapshot oder einen Klon aufgenommen, wenn sie in der Anwendungsdefinition explizit referenziert werden oder wenn sie Verweise auf einen der Anwendungs-Namespaces enthalten.

Erstellen Sie einen Snapshot mithilfe eines CR

Schritte

1. Erstellen Sie die benutzerdefinierte Ressourcendatei (CR) und benennen Sie sie `trident-protect-snapshot-cr.yaml`.
2. Konfigurieren Sie in der von Ihnen erstellten Datei die folgenden Attribute:
   - **metadata.name**: (*Erforderlich*) Der Name dieser benutzerdefinierten Ressource; wählen Sie einen eindeutigen und sinnvollen Namen für Ihre Umgebung.
   - **spec.applicationRef**: Der Kubernetes-Name der Anwendung, für die ein Snapshot erstellt werden soll.
   - **spec.appVaultRef**: (*Erforderlich*) Der Name des AppVault, wo die Snapshot-Inhalte (Metadaten) gespeichert werden sollen.
   - **spec.reclaimPolicy**: (*Optional*) Definiert, was mit dem AppArchive eines Snapshots geschieht, wenn die Snapshot-CR gelöscht wird. Das bedeutet, dass selbst wenn auf `Retain` gesetzt, der Snapshot gelöscht wird. Gültige Optionen:
     - `Retain` (Standard)
     - `Delete`

     ```
     apiVersion: protect.trident.netapp.io/v1
     kind: Snapshot
     metadata:
       namespace: my-app-namespace
       name: my-cr-name
     spec:
       applicationRef: my-application
       appVaultRef: appvault-name
       reclaimPolicy: Delete
     ```
3. Nachdem Sie die `trident-protect-snapshot-cr.yaml` Datei mit den korrekten Werten gefüllt haben, wenden Sie die CR an:

   ```
   kubectl apply -f trident-protect-snapshot-cr.yaml
   ```

Erstellen Sie einen Snapshot mithilfe der CLI

Schritte

1. Erstellen Sie den Snapshot, und ersetzen Sie die Werte in Klammern durch Informationen aus Ihrer Umgebung. Beispiel:

   ```
   tridentctl-protect create snapshot <my_snapshot_name> --appvault
   <my_appvault_name> --app <name_of_app_to_snapshot> -n
   <application_namespace>
   ```

Erstellen Sie eine bedarfsgesteuerte Datensicherung

Sie können eine App jederzeit sichern.

HINWEIS	Clusterbezogene Ressourcen werden in eine Sicherung, einen Snapshot oder einen Klon aufgenommen, wenn sie in der Anwendungsdefinition explizit referenziert werden oder wenn sie Verweise auf einen der Anwendungs-Namespaces enthalten.

Bevor Sie beginnen

Stellen Sie sicher, dass die Gültigkeitsdauer des AWS-Sitzungstokens für alle länger laufenden s3-Backup-Vorgänge ausreichend ist. Wenn das Token während des Backup-Vorgangs abläuft, kann der Vorgang fehlschlagen.

- Weitere Informationen zum Prüfen des Ablaufs des aktuellen Sitzungstokens finden Sie in der "AWS API-Dokumentation".
- Weitere Informationen zu Anmeldeinformationen für AWS-Ressourcen finden Sie in der "AWS IAM-Dokumentation".

Erstellen Sie eine Sicherung mithilfe eines CR

Schritte

1. Erstellen Sie die benutzerdefinierte Ressourcendatei (CR) und benennen Sie sie `trident-protect-backup-cr.yaml`.
2. Konfigurieren Sie in der von Ihnen erstellten Datei die folgenden Attribute:
   - **metadata.name**: (*Erforderlich*) Der Name dieser benutzerdefinierten Ressource; wählen Sie einen eindeutigen und sinnvollen Namen für Ihre Umgebung.
   - **spec.applicationRef**: (*Erforderlich*) Der Kubernetes-Name der zu sichernden Anwendung.
   - **spec.appVaultRef**: (*Erforderlich*) Der Name des AppVault, wo die Sicherungsinhalte gespeichert werden sollen.
   - **spec.dataMover**: (*Optional*) Eine Zeichenkette, die angibt, welches Sicherungstool für den Sicherungsvorgang verwendet werden soll. Mögliche Werte (Groß-/Kleinschreibung):
     - `Restic`
     - `Kopia` (Standard)
   - **spec.reclaimPolicy**: (*Optional*) Definiert, was mit einem Backup geschieht, wenn es aus seinem Anspruch entfernt wird. Mögliche Werte:
     - `Delete`
     - `Retain` (Standard)
   - **spec.snapshotRef**: (*Optional*): Name des Snapshots, der als Quelle für das Backup verwendet werden soll. Wenn kein Name angegeben wird, wird ein temporärer Snapshot erstellt und gesichert.

     Beispiel YAML:

```yaml
---
apiVersion: protect.trident.netapp.io/v1
kind: Backup
metadata:
  namespace: my-app-namespace
  name: my-cr-name
spec:
  applicationRef: my-application
  appVaultRef: appvault-name
  dataMover: Kopia
```

3. Nachdem Sie die `trident-protect-backup-cr.yaml` Datei mit den korrekten Werten gefüllt haben, wenden Sie die CR an:

```
kubectl apply -f trident-protect-backup-cr.yaml
```

Erstellen Sie ein Backup mithilfe der CLI

Schritte

1. Erstellen Sie die Sicherungskopie, indem Sie die Werte in Klammern durch Informationen aus Ihrer Umgebung ersetzen. Beispiel:

```
tridentctl-protect create backup <my_backup_name> --appvault <my-
vault-name> --app <name_of_app_to_back_up> --data-mover
<Kopia_or_Restic> -n <application_namespace>
```

   Sie können optional das `--full-backup` Flag verwenden, um anzugeben, ob eine Sicherung nicht-inkrementell sein soll. Standardmäßig sind alle Backups inkrementell. Wenn dieses Flag verwendet wird, wird die Sicherung nicht-inkrementell. Es ist Best Practice, regelmäßig eine vollständige Sicherung durchzuführen und dazwischen inkrementelle Sicherungen zu erstellen, um das Risiko bei der Wiederherstellung zu minimieren.

Unterstützte Sicherungsanmerkungen

Die folgende Tabelle beschreibt die Anmerkungen, die Sie beim Erstellen eines Backup-CR verwenden können:

Anmerkung	Typ	Beschreibung	Standardwert
protect.trident.netapp.io/full-backup	Zeichenkette	Legt fest, ob eine Sicherung nicht inkrementell sein soll. Setzen Sie auf `true`, um eine nicht inkrementelle Sicherung zu erstellen. Es ist Best Practice, regelmäßig eine vollständige Sicherung durchzuführen und dazwischen inkrementelle Sicherungen zu erstellen, um das Risiko bei der Wiederherstellung zu minimieren.	"false"
protect.trident.netapp.io/snapshot-completion-timeout	Zeichenkette	Die maximal zulässige Zeit für den Abschluss des gesamten Snapshot-Vorgangs.	"60m"
protect.trident.netapp.io/volume-snapshots-ready-to-use-timeout	Zeichenkette	Die maximal zulässige Zeitspanne, bis Volume-Snapshots den einsatzbereiten Zustand erreichen.	"30m"
protect.trident.netapp.io/volume-snapshots-created-timeout	Zeichenkette	Die maximal zulässige Zeit für die Erstellung von Volume-Snapshots.	"5m"
protect.trident.netapp.io/pvc-bind-timeout-sec	Zeichenkette	Maximale Zeit (in Sekunden), die auf neu erstellte PersistentVolumeClaims (PVCs) gewartet wird, um die `Bound` Phase zu erreichen, bevor der Vorgang fehlschlägt.	"1200" (20 Minuten)

Erstellen Sie einen Zeitplan für die Datensicherung

Eine Datensicherungsstrategie schützt eine App, indem sie Snapshots, Backups oder beides nach einem definierten Zeitplan erstellt. Sie können wählen, Snapshots und Backups stündlich, täglich, wöchentlich und monatlich zu erstellen, und Sie können die Anzahl der aufzubewahrenden Kopien angeben. Sie können ein nicht-inkrementelles vollständiges Backup mit der Annotation full-backup-rule planen. Standardmäßig sind alle Backups inkrementell. Das periodische Durchführen eines vollständigen Backups zusammen mit

inkrementellen Backups dazwischen hilft, das Risiko bei Wiederherstellungen zu reduzieren.

HINWEIS

- Sie können Zeitpläne für Snapshots nur erstellen, indem Sie `backupRetention` auf Null und `snapshotRetention` auf einen Wert größer als Null setzen. Das Setzen von `snapshotRetention` auf Null bedeutet, dass bei geplanten Backups zwar weiterhin Snapshots erstellt werden, diese jedoch temporär sind und unmittelbar nach Abschluss des Backups gelöscht werden.
- Clusterbezogene Ressourcen werden in eine Sicherung, einen Snapshot oder einen Klon aufgenommen, wenn sie in der Anwendungsdefinition explizit referenziert werden oder wenn sie Verweise auf einen der Anwendungs-Namespaces enthalten.

Erstellen Sie einen Zeitplan mithilfe eines CR

Schritte

1. Erstellen Sie die benutzerdefinierte Ressourcendatei (CR) und benennen Sie sie `trident-protect-schedule-cr.yaml`.
2. Konfigurieren Sie in der von Ihnen erstellten Datei die folgenden Attribute:
   - **metadata.name**: (*Erforderlich*) Der Name dieser benutzerdefinierten Ressource; wählen Sie einen eindeutigen und sinnvollen Namen für Ihre Umgebung.
   - **spec.dataMover**: (*Optional*) Eine Zeichenkette, die angibt, welches Sicherungstool für den Sicherungsvorgang verwendet werden soll. Mögliche Werte (Groß-/Kleinschreibung):
     - `Restic`
     - `Kopia` (Standard)
   - **spec.applicationRef**: Der Kubernetes-Name der zu sichernden Anwendung.
   - **spec.appVaultRef**: (*Erforderlich*) Der Name des AppVault, wo die Sicherungsinhalte gespeichert werden sollen.
   - **spec.backupRetention**: (*Erforderlich*) Die Anzahl der aufzubewahrenden Backups. Null bedeutet, dass keine Backups erstellt werden sollen (nur Snapshots).
   - **spec.backupReclaimPolicy**: (*Optional*) Legt fest, was mit einem Backup geschieht, wenn die Backup-CR während der Aufbewahrungsfrist gelöscht wird. Nach Ablauf der Aufbewahrungsfrist werden Backups immer gelöscht. Mögliche Werte (Groß-/Kleinschreibung):
     - `Retain` (Standard)
     - `Delete`
   - **spec.snapshotRetention**: (*Erforderlich*) Die Anzahl der aufzubewahrenden Snapshots. Zero bedeutet, dass keine Snapshots erstellt werden sollen.
   - **spec.snapshotReclaimPolicy**: (*Optional*) Legt fest, was mit einem Snapshot geschieht, wenn die Snapshot-CR während seiner Aufbewahrungsfrist gelöscht wird. Nach Ablauf der Aufbewahrungsfrist werden Snapshots immer gelöscht. Mögliche Werte (Groß-/Kleinschreibung):
     - `Retain`
     - `Delete` (Standard)
   - **spec.granularity**: Die Häufigkeit, mit der der Zeitplan ausgeführt werden soll. Mögliche Werte sowie zugehörige Pflichtfelder:
     - `Hourly` (erfordert, dass Sie `spec.minute` angeben)
     - `Daily` (erfordert, dass Sie `spec.minute` und `spec.hour` angeben)
     - `Weekly` (erfordert, dass Sie `spec.minute, spec.hour` angeben, und `spec.dayOfWeek`)
     - `Monthly` (erfordert, dass Sie `spec.minute, spec.hour` angeben, und `spec.dayOfMonth`)
     - `Custom`
   - **spec.dayOfMonth**: (*Optional*) Der Tag des Monats (1 - 31), an dem der Zeitplan ausgeführt werden soll. Dieses Feld ist erforderlich, wenn die Granularität auf `Monthly` eingestellt ist. Der Wert muss als Zeichenkette angegeben werden.

- **spec.dayOfWeek**: (*Optional*) Der Wochentag (0 - 7), an dem der Zeitplan ausgeführt werden soll. Werte von 0 oder 7 bedeuten Sonntag. Dieses Feld ist erforderlich, wenn die Granularität auf `Weekly` eingestellt ist. Der Wert muss als Zeichenkette angegeben werden.
- **spec.hour**: (*Optional*) Die Stunde des Tages (0–23), zu der der Zeitplan ausgeführt werden soll. Dieses Feld ist erforderlich, wenn die Granularität auf `Daily`, `Weekly` oder `Monthly` eingestellt ist. Der Wert muss als Zeichenkette angegeben werden.
- **spec.minute**: (*Optional*) Die Minute der Stunde (0–59), zu der der Zeitplan ausgeführt werden soll. Dieses Feld ist erforderlich, wenn die Granularität auf `Hourly`, `Daily`, `Weekly` oder `Monthly` gesetzt ist. Der Wert muss als Zeichenkette angegeben werden.
- **spec.runImmediately**: (*Optional*) Auf `true` setzen, um beim Erstellen des Zeitplans einen einmaligen, sofortigen Baseline-Lauf (Backup und/oder Snapshot gemäß den Aufbewahrungseinstellungen) auszulösen. Standardmäßig `false`. Dies ändert die nachfolgende Wiederholung nicht.

  Beispiel-YAML für Backup- und Snapshot-Zeitplan:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: Schedule
metadata:
  namespace: my-app-namespace
  name: my-cr-name
spec:
  dataMover: Kopia
  applicationRef: my-application
  appVaultRef: appvault-name
  backupRetention: "15"
  snapshotRetention: "15"
  granularity: Daily
  hour: "0"
  minute: "0"
```

  Beispiel-YAML für Snapshot-only-Zeitplan:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: Schedule
metadata:
  namespace: my-app-namespace
  name: my-snapshot-schedule
spec:
  applicationRef: my-application
  appVaultRef: appvault-name
  backupRetention: "0"
  snapshotRetention: "15"
  granularity: Daily
  hour: "2"
  minute: "0"
```

Beispiel-YAML für einen Zeitplan mit sofortiger Ausführung:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: Schedule
metadata:
  namespace: my-app-namespace
  name: my-daily-schedule-run-immediately
spec:
  applicationRef: my-application
  appVaultRef: appvault-name
  backupRetention: "7"
  snapshotRetention: "7"
  granularity: Daily
  hour: "3"
  minute: "0"
  runImmediately: true
```

3. Nachdem Sie die `trident-protect-schedule-cr.yaml` Datei mit den korrekten Werten gefüllt haben, wenden Sie die CR an:

```
kubectl apply -f trident-protect-schedule-cr.yaml
```

Erstellen Sie einen Zeitplan mithilfe der CLI

Schritte

1. Erstellen Sie den Schutzzeitplan, indem Sie die Werte in Klammern durch Informationen aus Ihrer Umgebung ersetzen. Beispiel:

HINWEIS Sie können `tridentctl-protect create schedule --help` verwenden, um detaillierte Hilfeinformationen zu diesem Befehl anzuzeigen.

```
tridentctl-protect create schedule <my_schedule_name> \
  --appvault <my_appvault_name> \
  --app <name_of_app_to_snapshot> \
  --backup-retention <how_many_backups_to_retain> \
  --backup-reclaim-policy <Retain|Delete (default Retain)> \
  --data-mover <Kopia_or_Restic> \
  --day-of-month <day_of_month_to_run_schedule> \
  --day-of-week <day_of_week_to_run_schedule> \
  --granularity <frequency_to_run> \
  --hour <hour_of_day_to_run> \
  --minute <minute_of_hour_to_run> \
  --recurrence-rule <recurrence> \
  --snapshot-retention <how_many_snapshots_to_retain> \
  --snapshot-reclaim-policy <Retain|Delete (default Delete)> \
  --full-backup-rule <string> \
  --run-immediately <true|false> \
  -n <application_namespace>
```

Die folgenden Optionen bieten zusätzliche Kontrolle über Ihren Zeitplan:

- **Planung vollständiger Backups**: Verwenden Sie das `--full-backup-rule`-Flag, um nicht-inkrementelle vollständige Backups zu planen. Dieses Flag funktioniert nur mit `--granularity Daily`. Mögliche Werte:
  - `Always`: Erstellen Sie jeden Tag ein vollständiges Backup.
  - Bestimmte Wochentage: Geben Sie einen oder mehrere Tage durch Kommas getrennt an (zum Beispiel `"Monday,Thursday"`). Gültige Werte: Montag, Dienstag, Mittwoch, Donnerstag, Freitag, Samstag, Sonntag.

  HINWEIS Die `--full-backup-rule` Kennzeichnung funktioniert nicht bei stündlicher, wöchentlicher oder monatlicher Granularität.

- **Sofortiger Basisdatenschutz**: Verwenden Sie `--run-immediately true`, um sofort bei der Erstellung des Zeitplans ein erstes Backup oder einen Snapshot zu erstellen, anstatt auf die erste geplante Ausführung zu warten. Standard ist `false`.
- **Snapshot-only-Zeitpläne**: Setzen Sie `--backup-retention 0` und geben Sie einen Wert größer als null für `--snapshot-retention` an.

Unterstützte Zeitplananmerkungen

Die folgende Tabelle beschreibt die Anmerkungen, die Sie beim Erstellen eines Schedule-CR verwenden können:

Anmerkung	Typ	Beschreibung	Standardwert
protect.trident.netapp.io/full-backup-rule	Zeichenkette	Legt die Regel für die Planung vollständiger Backups fest. Sie können es auf `Always` für konstante vollständige Sicherungen festlegen oder es basierend auf Ihren Anforderungen anpassen. Wenn Sie beispielsweise die tägliche Granularität wählen, können Sie die Wochentage angeben, an denen die vollständige Sicherung erfolgen soll (zum Beispiel `"Monday,Thursday"`). Gültige Wochentage sind: Montag, Dienstag, Mittwoch, Donnerstag, Freitag, Samstag, Sonntag. Beachten Sie, dass diese Annotation nur mit Zeitplänen verwendet werden kann, die `granularity` auf `Daily` gesetzt sind.	Nicht festgelegt (alle Backups sind inkrementell)
protect.trident.netapp.io/snapshot-completion-timeout	Zeichenkette	Die maximal zulässige Zeit für den Abschluss des gesamten Snapshot-Vorgangs.	"60m"
protect.trident.netapp.io/volume-snapshots-ready-to-use-timeout	Zeichenkette	Die maximal zulässige Zeitspanne, bis Volume-Snapshots den einsatzbereiten Zustand erreichen.	"30m"
protect.trident.netapp.io/volume-snapshots-created-timeout	Zeichenkette	Die maximal zulässige Zeit für die Erstellung von Volume-Snapshots.	"5m"
protect.trident.netapp.io/pvc-bind-timeout-sec	Zeichenkette	Maximale Zeit (in Sekunden), die auf neu erstellte PersistentVolumeClaims (PVCs) gewartet wird, um die `Bound` Phase zu erreichen, bevor der Vorgang fehlschlägt.	"1200" (20 Minuten)

Einen Snapshot löschen

Löschen Sie die geplanten oder On-Demand-Snapshots, die Sie nicht mehr benötigen.

Schritte

1. Entfernen Sie die mit dem Snapshot verknüpfte Snapshot-CR:

```
kubectl delete snapshot <snapshot_name> -n my-app-namespace
```

Ein Backup löschen

Löschen Sie die geplanten oder On-Demand-Backups, die Sie nicht mehr benötigen.

HINWEIS Stellen Sie sicher, dass die Rückgewinnungsrichtlinie auf `Delete` gesetzt ist, um alle Sicherungsdaten aus dem Objektspeicher zu entfernen. Die Standardeinstellung der Richtlinie ist `Retain`, um versehentlichen Datenverlust zu vermeiden. Wenn die Richtlinie nicht auf `Delete` geändert wird, verbleiben die Sicherungsdaten im Objektspeicher und müssen manuell gelöscht werden.

Schritte

1. Entfernen Sie die mit dem Backup verknüpfte Backup-CR:

```
kubectl delete backup <backup_name> -n my-app-namespace
```

Überprüfen Sie den Status eines Sicherungsvorgangs

Sie können die Befehlszeile verwenden, um den Status eines Sicherungsvorgangs zu überprüfen, der gerade ausgeführt wird, abgeschlossen ist oder fehlgeschlagen ist.

Schritte

1. Verwenden Sie den folgenden Befehl, um den Status des Sicherungsvorgangs abzurufen, und ersetzen Sie die Werte in Klammern durch Informationen aus Ihrer Umgebung:

```
kubectl get backup -n <namespace_name> <my_backup_cr_name> -o jsonpath
='{.status}'
```

Backup und Restore für azure-netapp-files (ANF)-Vorgänge aktivieren

Wenn Sie Trident Protect installiert haben, können Sie die platzsparende Sicherungs- und Wiederherstellungsfunktion für Speicher-Backends aktivieren, die die Speicherklasse azure-netapp-files verwenden und vor Trident 24.06 erstellt wurden. Diese Funktionalität funktioniert mit NFSv4-Volumes und verbraucht keinen zusätzlichen Speicherplatz aus dem Kapazitätspool.

Bevor Sie beginnen

Stellen Sie Folgendes sicher:

- Sie haben Trident Protect installiert.
- Sie haben eine Anwendung in Trident Protect definiert. Diese Anwendung verfügt nur über eingeschränkte Schutzfunktionen, bis Sie dieses Verfahren abgeschlossen haben.
- Sie haben `azure-netapp-files` als Standard-Speicherklasse für Ihr Speicher-Backend ausgewählt.

Für Konfigurationsschritte erweitern

1. Führen Sie Folgendes in Trident aus, wenn das ANF-Volume vor dem Upgrade auf Trident 24.10 erstellt wurde:

   a. Aktivieren Sie das Snapshot-Verzeichnis für jedes PV, das auf azure-netapp-files basiert und der Anwendung zugeordnet ist:

   ```
   tridentctl update volume <pv name> --snapshot-dir=true -n trident
   ```

   b. Vergewissern Sie sich, dass das Snapshot-Verzeichnis für jedes zugehörige PV aktiviert wurde:

   ```
   tridentctl get volume <pv name> -n trident -o yaml | grep
   snapshotDir
   ```

   Antwort:

   ```
   snapshotDirectory: "true"
   ```

   +
   Wenn das Snapshot-Verzeichnis nicht aktiviert ist, wählt Trident Protect die reguläre Sicherungsfunktion, die während des Sicherungsvorgangs temporär Speicherplatz im Kapazitätspool belegt. Stellen Sie in diesem Fall sicher, dass im Kapazitätspool ausreichend Speicherplatz vorhanden ist, um ein temporäres Volume in der Größe des zu sichernden Volumes zu erstellen.

Ergebnis

Die Anwendung ist für Backup und Restore mit Trident Protect bereit. Jede PVC kann auch von anderen Anwendungen für Backups und Restores verwendet werden.

Anwendungen wiederherstellen

Anwendungen mit Trident Protect wiederherstellen

Mit Trident Protect können Sie Ihre Anwendung aus einem Snapshot oder einer Sicherung wiederherstellen. Die Wiederherstellung aus einem vorhandenen Snapshot ist schneller, wenn die Anwendung im selben Cluster wiederhergestellt wird.

HINWEIS

- Wenn Sie eine Anwendung wiederherstellen, werden alle für die Anwendung konfigurierten Ausführungs-Hooks mit der Anwendung wiederhergestellt. Wenn ein Ausführungs-Hook nach der Wiederherstellung vorhanden ist, wird er automatisch als Teil des Wiederherstellungsvorgangs ausgeführt.
- Die Wiederherstellung aus einem Backup in einen anderen Namespace oder in den ursprünglichen Namespace wird für qtree Volumes unterstützt. Die Wiederherstellung aus einem Snapshot in einen anderen Namespace oder in den ursprünglichen Namespace wird für qtree Volumes jedoch nicht unterstützt.
- Sie können die Wiederherstellungsvorgänge mithilfe erweiterter Einstellungen anpassen. Weitere Informationen finden Sie unter "Verwenden Sie die erweiterten Trident Protect-Wiederherstellungseinstellungen".

Wiederherstellung aus einem Backup in einen anderen Namensraum

Wenn Sie eine Sicherung mithilfe einer BackupRestore CR in einem anderen Namespace wiederherstellen, stellt Trident Protect die Anwendung in einem neuen Namespace wieder her und erstellt eine Anwendungs-CR für die wiederhergestellte Anwendung. Um die wiederhergestellte Anwendung zu schützen, erstellen Sie bedarfsgesteuerte Backups oder Snapshots oder legen Sie einen Schutzzeitplan fest.

HINWEIS

- Die Wiederherstellung einer Sicherung in einem anderen Namensraum mit vorhandenen Ressourcen ändert keine Ressourcen, die denselben Namen wie die in der Sicherung haben. Um alle Ressourcen in der Sicherung wiederherzustellen, löschen und erstellen Sie entweder den Zielnamensraum neu oder stellen Sie die Sicherung in einem neuen Namensraum wieder her.
- Wenn Sie eine CR zur Wiederherstellung in einem neuen Namespace verwenden, müssen Sie den Ziel-Namespace manuell erstellen, bevor Sie die CR anwenden. Trident Protect erstellt Namespaces automatisch nur bei Verwendung der CLI.

Bevor Sie beginnen

Stellen Sie sicher, dass die Gültigkeitsdauer des AWS-Sitzungstokens für alle länger dauernden s3-Wiederherstellungsvorgänge ausreichend ist. Wenn das Token während des Wiederherstellungsvorgangs abläuft, kann der Vorgang fehlschlagen.

- Weitere Informationen zum Prüfen des Ablaufs des aktuellen Sitzungstokens finden Sie in der "AWS API-Dokumentation".
- Weitere Informationen zu Anmeldeinformationen für AWS-Ressourcen finden Sie in der "AWS IAM-Dokumentation".

HINWEIS

Wenn Sie Backups mit Kopia als Data Mover wiederherstellen, können Sie optional Anmerkungen in der CR oder über die CLI angeben, um das Verhalten des von Kopia verwendeten temporären Speichers zu steuern. Weitere Informationen über die Optionen, die Sie konfigurieren können, finden Sie in der "Kopia-Dokumentation". Verwenden Sie den `tridentctl-protect create --help`-Befehl, um weitere Informationen zum Angeben von Anmerkungen mit der Trident Protect CLI zu erhalten.

Verwenden Sie einen CR

Schritte

1. Erstellen Sie die benutzerdefinierte Ressourcendatei (CR) und benennen Sie sie `trident-protect-backup-restore-cr.yaml`.
2. Konfigurieren Sie in der von Ihnen erstellten Datei die folgenden Attribute:
   - **metadata.name**: (*Erforderlich*) Der Name dieser benutzerdefinierten Ressource; wählen Sie einen eindeutigen und sinnvollen Namen für Ihre Umgebung.
   - **spec.appArchivePath**: Der Pfad innerhalb von AppVault, in dem die Sicherungsinhalte gespeichert sind. Sie können den folgenden Befehl verwenden, um diesen Pfad zu finden:

     ```
     kubectl get backups <BACKUP_NAME> -n my-app-namespace -o
     jsonpath='{.status.appArchivePath}'
     ```

   - **spec.appVaultRef**: (*Erforderlich*) Der Name des AppVault, in dem die Sicherungsinhalte gespeichert sind.
   - **spec.destinationApplicationName**: (*Optional*) Der Name für die wiederhergestellte Anwendung. Falls angegeben, verwendet die wiederhergestellte Anwendung diesen Namen. Falls nicht angegeben, verwendet die wiederhergestellte Anwendung den Namen der Quellanwendung.
   - **spec.namespaceMapping**: Die Zuordnung des Quell-Namespace des Wiederherstellungsvorgangs zum Ziel-Namespace. Ersetzen Sie `my-source-namespace` und `my-destination-namespace` durch Informationen aus Ihrer Umgebung.

     ```
     ---
     apiVersion: protect.trident.netapp.io/v1
     kind: BackupRestore
     metadata:
       name: my-cr-name
       namespace: my-destination-namespace
     spec:
       appArchivePath: my-backup-path
       appVaultRef: appvault-name
       destinationApplicationName: my-new-app-name
       namespaceMapping: [{"source": "my-source-namespace",
     "destination": "my-destination-namespace"}]
     ```

3. (*Optional*) Falls Sie nur bestimmte Ressourcen der Anwendung für die Wiederherstellung auswählen möchten, fügen Sie Filter hinzu, die Ressourcen mit bestimmten Bezeichnungen ein- oder ausschließen:

HINWEIS	Trident Protect wählt bestimmte Ressourcen automatisch aus, weil sie mit den von Ihnen ausgewählten Ressourcen in Beziehung stehen. Wenn Sie beispielsweise eine Ressource für einen persistenten Volume-Claim auswählen und diese einen zugehörigen Pod hat, wird Trident Protect auch den zugehörigen Pod wiederherstellen.

- **resourceFilter.resourceSelectionCriteria**: (Für die Filterung erforderlich) Verwenden Sie `Include` oder `Exclude`, um eine in resourceMatchers definierte Ressource ein- oder auszuschließen. Fügen Sie die folgenden resourceMatchers-Parameter hinzu, um die Ressourcen zu definieren, die ein- oder auszuschließen sind:
  - **resourceFilter.resourceMatchers**: Ein Array von resourceMatcher-Objekten. Wenn Sie mehrere Elemente in diesem Array definieren, werden diese mit einer ODER-Verknüpfung verglichen und die Felder innerhalb jedes Elements (group, kind, version) werden mit einer UND-Verknüpfung verglichen.
    - **resourceMatchers[].group**: (*Optional*) Gruppe der zu filternden Ressource.
    - **resourceMatchers[].kind**: (*Optional*) Art der zu filternden Ressource.
    - **resourceMatchers[].version**: (*Optional*) Version der zu filternden Ressource.
    - **resourceMatchers[].names**: (*Optional*) Namen im Kubernetes metadata.name-Feld der Ressource, die gefiltert werden soll.
    - **resourceMatchers[].namespaces**: (*Optional*) Namespaces im Kubernetes metadata.name-Feld der Ressource, die gefiltert werden soll.
    - **resourceMatchers[].labelSelectors**: (*Optional*) Label-Selektorzeichenfolge im Kubernetes-Metadatenfeld name der Ressource, wie definiert in der "Kubernetes-Dokumentation". Zum Beispiel: `"trident.netapp.io/os=linux"`.

      Beispiel:

```
spec:
  resourceFilter:
    resourceSelectionCriteria: "Include"
    resourceMatchers:
      - group: my-resource-group-1
        kind: my-resource-kind-1
        version: my-resource-version-1
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
      - group: my-resource-group-2
        kind: my-resource-kind-2
        version: my-resource-version-2
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
```

4. Nachdem Sie die `trident-protect-backup-restore-cr.yaml` Datei mit den korrekten Werten gefüllt haben, wenden Sie die CR an:

```
kubectl apply -f trident-protect-backup-restore-cr.yaml
```

Verwenden Sie die Befehlszeile

Schritte

1. Stellen Sie die Sicherung in einem anderen Namensraum wieder her, indem Sie die Werte in Klammern durch Informationen aus Ihrer Umgebung ersetzen. Das `namespace-mapping` Argument verwendet durch Doppelpunkte getrennte Namensräume, um Quell-Namensräume den korrekten Ziel-Namensräumen im Format `source1:dest1,source2:dest2` zuzuordnen. Beispiel:

```
tridentctl-protect create backuprestore <my_restore_name> \
--backup <backup_namespace>/<backup_to_restore> \
--namespace-mapping <source_to_destination_namespace_mapping> \
--destination-app-name<custom_app_name>\
-n <application_namespace>
```

Stellen Sie aus einer Sicherung in den ursprünglichen Namensraum wieder her

Sie können eine Sicherung jederzeit im ursprünglichen Namensraum wiederherstellen. Wenn Sie eine Wiederherstellung vor Ort durchführen, verwaltet Trident Protect automatisch Schutzzeitpläne und laufende Vorgänge, um ungültige Wiederherstellungspunkte zu verhindern:

- Alle für die Anwendung aktivierten Schutzzeitpläne werden vor Beginn der Wiederherstellung deaktiviert. Dadurch wird verhindert, dass geplante Sicherungen oder Snapshots ausgeführt werden, während die Anwendungsressourcen wiederhergestellt werden.
- Nach erfolgreicher Wiederherstellung werden nur die vor der Wiederherstellung aktivierten Zeitpläne wieder aktiviert. Bereits deaktivierte Zeitpläne bleiben deaktiviert.
- Laufende Sicherungs- oder Snapshot-Vorgänge werden vor Beginn der Wiederherstellung abgebrochen. Wird ein Vorgang nicht innerhalb von 5 Minuten abgebrochen, wird die Wiederherstellung fortgesetzt und eine Warnung im Wiederherstellungs-CR-Status protokolliert.

Bevor Sie beginnen

Stellen Sie sicher, dass die Gültigkeitsdauer des AWS-Sitzungstokens für alle länger dauernden s3-Wiederherstellungsvorgänge ausreichend ist. Wenn das Token während des Wiederherstellungsvorgangs abläuft, kann der Vorgang fehlschlagen.

- Weitere Informationen zum Prüfen des Ablaufs des aktuellen Sitzungstokens finden Sie in der "AWS API-Dokumentation".
- Weitere Informationen zu Anmeldeinformationen für AWS-Ressourcen finden Sie in der "AWS IAM-Dokumentation".

HINWEIS	Wenn Sie Backups mit Kopia als Data Mover wiederherstellen, können Sie optional Anmerkungen in der CR oder über die CLI angeben, um das Verhalten des von Kopia verwendeten temporären Speichers zu steuern. Weitere Informationen über die Optionen, die Sie konfigurieren können, finden Sie in der "Kopia-Dokumentation". Verwenden Sie den `tridentctl-protect create --help`-Befehl, um weitere Informationen zum Angeben von Anmerkungen mit der Trident Protect CLI zu erhalten.

Verwenden Sie einen CR

Schritte

1. Erstellen Sie die benutzerdefinierte Ressourcendatei (CR) und benennen Sie sie `trident-protect-backup-ipr-cr.yaml`.
2. Konfigurieren Sie in der von Ihnen erstellten Datei die folgenden Attribute:

   - **metadata.name**: (*Erforderlich*) Der Name dieser benutzerdefinierten Ressource; wählen Sie einen eindeutigen und sinnvollen Namen für Ihre Umgebung.
   - **spec.appArchivePath**: Der Pfad innerhalb von AppVault, in dem die Sicherungsinhalte gespeichert sind. Sie können den folgenden Befehl verwenden, um diesen Pfad zu finden:

     ```
     kubectl get backups <BACKUP_NAME> -n my-app-namespace -o jsonpath='{.status.appArchivePath}'
     ```

   - **spec.appVaultRef**: (*Erforderlich*) Der Name des AppVault, in dem die Sicherungsinhalte gespeichert sind.

     Beispiel:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: BackupInplaceRestore
metadata:
  name: my-cr-name
  namespace: my-app-namespace
spec:
  appArchivePath: my-backup-path
  appVaultRef: appvault-name
```

3. (*Optional*) Falls Sie nur bestimmte Ressourcen der Anwendung für die Wiederherstellung auswählen möchten, fügen Sie Filter hinzu, die Ressourcen mit bestimmten Bezeichnungen ein- oder ausschließen:

HINWEIS	Trident Protect wählt bestimmte Ressourcen automatisch aus, weil sie mit den von Ihnen ausgewählten Ressourcen in Beziehung stehen. Wenn Sie beispielsweise eine Ressource für einen persistenten Volume-Claim auswählen und diese einen zugehörigen Pod hat, wird Trident Protect auch den zugehörigen Pod wiederherstellen.

   - **resourceFilter.resourceSelectionCriteria**: (Für die Filterung erforderlich) Verwenden Sie `Include` oder `Exclude`, um eine in resourceMatchers definierte Ressource ein- oder auszuschließen. Fügen Sie die folgenden resourceMatchers-Parameter hinzu, um die Ressourcen zu definieren, die ein- oder auszuschließen sind:
     - **resourceFilter.resourceMatchers**: Ein Array von resourceMatcher-Objekten. Wenn Sie mehrere Elemente in diesem Array definieren, werden diese mit einer ODER-Verknüpfung verglichen und die Felder innerhalb jedes Elements (group, kind, version) werden mit einer

UND-Verknüpfung verglichen.

- **resourceMatchers[].group**: (*Optional*) Gruppe der zu filternden Ressource.
- **resourceMatchers[].kind**: (*Optional*) Art der zu filternden Ressource.
- **resourceMatchers[].version**: (*Optional*) Version der zu filternden Ressource.
- **resourceMatchers[].names**: (*Optional*) Namen im Kubernetes metadata.name-Feld der Ressource, die gefiltert werden soll.
- **resourceMatchers[].namespaces**: (*Optional*) Namespaces im Kubernetes metadata.name-Feld der Ressource, die gefiltert werden soll.
- **resourceMatchers[].labelSelectors**: (*Optional*) Label-Selektorzeichenfolge im Kubernetes-Metadatenfeld name der Ressource, wie definiert in der "Kubernetes-Dokumentation". Zum Beispiel: `"trident.netapp.io/os=linux"`.

  Beispiel:

```
spec:
  resourceFilter:
    resourceSelectionCriteria: "Include"
    resourceMatchers:
      - group: my-resource-group-1
        kind: my-resource-kind-1
        version: my-resource-version-1
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
      - group: my-resource-group-2
        kind: my-resource-kind-2
        version: my-resource-version-2
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
```

4. Nachdem Sie die `trident-protect-backup-ipr-cr.yaml` Datei mit den korrekten Werten gefüllt haben, wenden Sie die CR an:

```
kubectl apply -f trident-protect-backup-ipr-cr.yaml
```

Verwenden Sie die Befehlszeile

Schritte

1. Stellen Sie die Sicherung im ursprünglichen Namensraum wieder her, indem Sie die Werte in Klammern durch Informationen aus Ihrer Umgebung ersetzen. Das `backup` Argument verwendet einen Namensraum und einen Sicherungsnamen im Format `<namespace>/<name>`. Beispiel:

```
tridentctl-protect create backupinplacerestore <my_restore_name> \
--backup <namespace/backup_to_restore> \
-n <application_namespace>
```

Wiederherstellung aus einem Backup auf einem anderen Cluster

Sie können ein Backup auf einem anderen Cluster wiederherstellen, wenn es ein Problem mit dem ursprünglichen Cluster gibt.

HINWEIS

- Wenn Sie Backups mit Kopia als Data Mover wiederherstellen, können Sie optional Anmerkungen in der CR oder über die CLI angeben, um das Verhalten des von Kopia verwendeten temporären Speichers zu steuern. Weitere Informationen über die Optionen, die Sie konfigurieren können, finden Sie in der "Kopia-Dokumentation". Verwenden Sie den `tridentctl-protect create --help`-Befehl, um weitere Informationen zum Angeben von Anmerkungen mit der Trident Protect CLI zu erhalten.
- Wenn Sie eine CR zur Wiederherstellung in einem neuen Namespace verwenden, müssen Sie den Ziel-Namespace manuell erstellen, bevor Sie die CR anwenden. Trident Protect erstellt Namespaces automatisch nur bei Verwendung der CLI.

Bevor Sie beginnen

Stellen Sie sicher, dass die folgenden Voraussetzungen erfüllt sind:

- Auf dem Ziel-Cluster ist Trident Protect installiert.
- Der Ziel-Cluster hat Zugriff auf den Bucket-Pfad desselben AppVault wie der Quell-Cluster, in dem die Sicherung gespeichert ist.
- Stellen Sie sicher, dass Ihre lokale Umgebung eine Verbindung zum im AppVault CR definierten Objektspeicher-Bucket herstellen kann, wenn Sie den `tridentctl-protect get appvaultcontent` Befehl ausführen. Wenn Netzwerkbeschränkungen den Zugriff verhindern, führen Sie die Trident Protect CLI stattdessen innerhalb eines Pods auf dem Ziel-Cluster aus.
- Stellen Sie sicher, dass die Gültigkeitsdauer des AWS-Sitzungstokens für alle länger dauernden Wiederherstellungsvorgänge ausreichend ist. Wenn das Token während des Wiederherstellungsvorgangs abläuft, kann der Vorgang fehlschlagen.
  - Weitere Informationen zum Prüfen des Ablaufs des aktuellen Sitzungstokens finden Sie in der "AWS API-Dokumentation".
  - Weitere Informationen zu Anmeldeinformationen für AWS-Ressourcen finden Sie in der "AWS-Dokumentation".

Schritte

1. Überprüfen Sie, ob die AppVault CR auf dem Ziel-Cluster mithilfe des Trident Protect CLI-Plugins vorhanden ist:

```
tridentctl-protect get appvault --context <destination_cluster_name>
```

HINWEIS	Falls die AppVault CR auf dem Ziel-Cluster nicht existiert, erstellen Sie sie gemäß den Schritten in "Verwenden Sie Trident Protect AppVault-Objekte, um Buckets zu verwalten".

2. Sehen Sie sich die Sicherungsinhalte des verfügbaren AppVault auf dem Ziel-Cluster an und notieren Sie sich `appArchivePath` der Sicherung, die Sie wiederherstellen möchten:

```
tridentctl-protect get appvaultcontent <appvault_name> \
--show-resources backup \
--show-paths \
--context <destination_cluster_name>
```

Durch Ausführen dieses Befehls werden die verfügbaren Backups im AppVault angezeigt, einschließlich ihrer Ursprungscluster, entsprechenden Anwendungsnamen, Zeitstempel und Archivpfade.

Beispielausgabe:

```
+-------------+-----------+--------+-----------------
+--------------------------+-------------+
|   CLUSTER   |    APP    |  TYPE  |      NAME       |         TIMESTAMP
|    PATH     |
+-------------+-----------+--------+-----------------
+--------------------------+-------------+
| production1 | wordpress | backup | wordpress-bkup-1| 2024-10-30
08:37:40 (UTC)| backuppath1 |
| production1 | wordpress | backup | wordpress-bkup-2| 2024-10-30
08:37:40 (UTC)| backuppath2 |
+-------------+-----------+--------+-----------------
+--------------------------+-------------+
```

3. Stellen Sie die Anwendung im Ziel-Cluster mithilfe des AppVault-Namens und des Archivpfads wieder her:

HINWEIS	Bei Verwendung eines CR muss sichergestellt werden, dass der für die Anwendungswiederherstellung vorgesehene Namespace auf dem Ziel-Cluster existiert.

Verwenden Sie einen CR

1. Erstellen Sie die benutzerdefinierte Ressourcendatei (CR) und benennen Sie sie `trident-protect-backup-restore-cr.yaml`.
2. Konfigurieren Sie in der von Ihnen erstellten Datei die folgenden Attribute:
   - **metadata.name**: (*Erforderlich*) Der Name dieser benutzerdefinierten Ressource; wählen Sie einen eindeutigen und sinnvollen Namen für Ihre Umgebung.
   - **spec.appVaultRef**: (*Erforderlich*) Der Name des AppVault, in dem die Sicherungsinhalte gespeichert sind.
   - **spec.appArchivePath**: (*Erforderlich*) Der Pfad innerhalb von AppVault, in dem die Sicherungsinhalte gespeichert sind. Verwenden Sie den Befehl aus Schritt 2, um die Sicherungsinhalte anzuzeigen und `appArchivePath`für die Sicherung zu finden, die Sie wiederherstellen möchten.
   - **spec.destinationApplicationName**: (*Optional*) Der Name für die wiederhergestellte Anwendung. Falls angegeben, verwendet die wiederhergestellte Anwendung diesen Namen. Falls nicht angegeben, verwendet die wiederhergestellte Anwendung den Namen der Quellanwendung.
   - **spec.namespaceMapping**: Die Zuordnung des Quell-Namespace des Wiederherstellungsvorgangs zum Ziel-Namespace. Ersetzen Sie `my-source-namespace` und `my-destination-namespace` durch Informationen aus Ihrer Umgebung.

     Beispiel:

```
apiVersion: protect.trident.netapp.io/v1
kind: BackupRestore
metadata:
  name: my-cr-name
  namespace: my-destination-namespace
spec:
  appVaultRef: appvault-name
  appArchivePath: my-backup-path
  destinationApplicationName: my-new-app-name
  namespaceMapping: [{"source": "my-source-namespace", "
destination": "my-destination-namespace"}]
```

3. Nachdem Sie die `trident-protect-backup-restore-cr.yaml` Datei mit den korrekten Werten gefüllt haben, wenden Sie die CR an:

```
kubectl apply -f trident-protect-backup-restore-cr.yaml
```

Verwenden Sie die Befehlszeile

1. Verwenden Sie den folgenden Befehl, um die Anwendung wiederherzustellen, und ersetzen Sie die Werte in Klammern durch Informationen aus Ihrer Umgebung. Das Argument namespace-mapping verwendet durch Doppelpunkte getrennte Namensräume, um Quell-Namensräume den korrekten Ziel-Namensräumen im Format Quelle1:Ziel1,Quelle2:Ziel2 zuzuordnen. Beispiel:

```
tridentctl-protect create backuprestore <restore_name> \
--namespace-mapping <source_to_destination_namespace_mapping> \
--appvault <appvault_name> \
--path <backup_path> \
--destination-app-name <custom_app_name> \
--context <destination_cluster_name> \
-n <application_namespace>
```

Wiederherstellung aus einem Snapshot in einen anderen Namespace

Sie können Daten aus einem Snapshot mithilfe einer benutzerdefinierten Ressource (CR) entweder in einem anderen Namespace oder im ursprünglichen Quell-Namespace wiederherstellen. Wenn Sie einen Snapshot mithilfe einer SnapshotRestore CR in einem anderen Namespace wiederherstellen, stellt Trident Protect die Anwendung in einem neuen Namespace wieder her und erstellt eine Anwendungs-CR für die wiederhergestellte Anwendung. Um die wiederhergestellte Anwendung zu schützen, erstellen Sie bedarfsgesteuerte Backups oder Snapshots oder legen Sie einen Schutzzeitplan fest.

HINWEIS

- SnapshotRestore unterstützt das `spec.storageClassMapping` Attribut, jedoch nur, wenn die Quell- und Ziel-Speicherklassen dasselbe Speicher-Backend verwenden. Wenn Sie versuchen, auf eine `StorageClass` wiederherzustellen, die ein anderes Speicher-Backend verwendet, schlägt der Wiederherstellungsvorgang fehl.
- Wenn Sie eine CR zur Wiederherstellung in einem neuen Namespace verwenden, müssen Sie den Ziel-Namespace manuell erstellen, bevor Sie die CR anwenden. Trident Protect erstellt Namespaces automatisch nur bei Verwendung der CLI.

Bevor Sie beginnen

Stellen Sie sicher, dass die Gültigkeitsdauer des AWS-Sitzungstokens für alle länger dauernden s3-Wiederherstellungsvorgänge ausreichend ist. Wenn das Token während des Wiederherstellungsvorgangs abläuft, kann der Vorgang fehlschlagen.

- Weitere Informationen zum Prüfen des Ablaufs des aktuellen Sitzungstokens finden Sie in der "AWS API-Dokumentation".
- Weitere Informationen zu Anmeldeinformationen für AWS-Ressourcen finden Sie in der "AWS IAM-Dokumentation".

Verwenden Sie einen CR

Schritte

1. Erstellen Sie die benutzerdefinierte Ressourcendatei (CR) und benennen Sie sie `trident-protect-snapshot-restore-cr.yaml`.
2. Konfigurieren Sie in der von Ihnen erstellten Datei die folgenden Attribute:
   - **metadata.name**: (*Erforderlich*) Der Name dieser benutzerdefinierten Ressource; wählen Sie einen eindeutigen und sinnvollen Namen für Ihre Umgebung.
   - **spec.appVaultRef**: (*Erforderlich*) Der Name des AppVault, in dem die Snapshot-Inhalte gespeichert sind.
   - **spec.appArchivePath**: Der Pfad innerhalb von AppVault, in dem die Snapshot-Inhalte gespeichert sind. Sie können den folgenden Befehl verwenden, um diesen Pfad zu finden:

     ```
     kubectl get snapshots <SNAPHOT_NAME> -n my-app-namespace -o jsonpath='{.status.appArchivePath}'
     ```

   - **spec.destinationApplicationName**: (*Optional*) Der Name für die wiederhergestellte Anwendung. Falls angegeben, verwendet die wiederhergestellte Anwendung diesen Namen. Falls nicht angegeben, verwendet die wiederhergestellte Anwendung den Namen der Quellanwendung.
   - **spec.namespaceMapping**: Die Zuordnung des Quell-Namespace des Wiederherstellungsvorgangs zum Ziel-Namespace. Ersetzen Sie `my-source-namespace` und `my-destination-namespace` durch Informationen aus Ihrer Umgebung.

     ```
     ---
     apiVersion: protect.trident.netapp.io/v1
     kind: SnapshotRestore
     metadata:
       name: my-cr-name
       namespace: my-app-namespace
     spec:
       appVaultRef: appvault-name
       appArchivePath: my-snapshot-path
       namespaceMapping: [{"source": "my-source-namespace",
     "destination": "my-destination-namespace"}]
     ```

3. (*Optional*) Falls Sie nur bestimmte Ressourcen der Anwendung für die Wiederherstellung auswählen möchten, fügen Sie Filter hinzu, die Ressourcen mit bestimmten Bezeichnungen ein- oder ausschließen:

HINWEIS	Trident Protect wählt bestimmte Ressourcen automatisch aus, weil sie mit den von Ihnen ausgewählten Ressourcen in Beziehung stehen. Wenn Sie beispielsweise eine Ressource für einen persistenten Volume-Claim auswählen und diese einen zugehörigen Pod hat, wird Trident Protect auch den zugehörigen Pod wiederherstellen.

- **resourceFilter.resourceSelectionCriteria**: (Für die Filterung erforderlich) Verwenden Sie `Include` oder `Exclude`, um eine in resourceMatchers definierte Ressource ein- oder auszuschließen. Fügen Sie die folgenden resourceMatchers-Parameter hinzu, um die Ressourcen zu definieren, die ein- oder auszuschließen sind:
  - **resourceFilter.resourceMatchers**: Ein Array von resourceMatcher-Objekten. Wenn Sie mehrere Elemente in diesem Array definieren, werden diese mit einer ODER-Verknüpfung verglichen und die Felder innerhalb jedes Elements (group, kind, version) werden mit einer UND-Verknüpfung verglichen.
    - **resourceMatchers[].group**: (*Optional*) Gruppe der zu filternden Ressource.
    - **resourceMatchers[].kind**: (*Optional*) Art der zu filternden Ressource.
    - **resourceMatchers[].version**: (*Optional*) Version der zu filternden Ressource.
    - **resourceMatchers[].names**: (*Optional*) Namen im Kubernetes metadata.name-Feld der Ressource, die gefiltert werden soll.
    - **resourceMatchers[].namespaces**: (*Optional*) Namespaces im Kubernetes metadata.name-Feld der Ressource, die gefiltert werden soll.
    - **resourceMatchers[].labelSelectors**: (*Optional*) Label-Selektorzeichenfolge im Kubernetes-Metadatenfeld name der Ressource, wie definiert in der "Kubernetes-Dokumentation". Zum Beispiel: `"trident.netapp.io/os=linux"`.

      Beispiel:

```
spec:
  resourceFilter:
    resourceSelectionCriteria: "Include"
    resourceMatchers:
      - group: my-resource-group-1
        kind: my-resource-kind-1
        version: my-resource-version-1
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
      - group: my-resource-group-2
        kind: my-resource-kind-2
        version: my-resource-version-2
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
```

4. Nachdem Sie die `trident-protect-snapshot-restore-cr.yaml` Datei mit den korrekten Werten gefüllt haben, wenden Sie die CR an:

```
kubectl apply -f trident-protect-snapshot-restore-cr.yaml
```

Verwenden Sie die Befehlszeile

Schritte

1. Stellen Sie den Snapshot in einem anderen Namespace wieder her, wobei Sie die Werte in Klammern durch Informationen aus Ihrer Umgebung ersetzen.

   - Das `snapshot` Argument verwendet einen Namespace und einen Snapshot-Namen im Format `<namespace>/<name>`.
   - Das `namespace-mapping` Argument verwendet durch Doppelpunkte getrennte Namensräume, um Quell-Namensräume den korrekten Ziel-Namensräumen im Format `source1:dest1,source2:dest2` zuzuordnen.

     Beispiel:

```
tridentctl-protect create snapshotrestore <my_restore_name> \
--snapshot <namespace/snapshot_to_restore> \
--namespace-mapping <source_to_destination_namespace_mapping> \
--destination-app-name <custom_app_name> \
-n <application_namespace>
```

Wiederherstellung aus einem Snapshot in den ursprünglichen Namensraum

Sie können einen Snapshot jederzeit im ursprünglichen Namensraum wiederherstellen. Wenn Sie eine Wiederherstellung vor Ort durchführen, verwaltet Trident Protect automatisch Schutzzeitpläne und laufende Vorgänge, um ungültige Wiederherstellungspunkte zu verhindern:

- Alle für die Anwendung aktivierten Schutzzeitpläne werden vor Beginn der Wiederherstellung deaktiviert. Dadurch wird verhindert, dass geplante Sicherungen oder Snapshots ausgeführt werden, während die Anwendungsressourcen wiederhergestellt werden.
- Nach erfolgreicher Wiederherstellung werden nur die vor der Wiederherstellung aktivierten Zeitpläne wieder aktiviert. Bereits deaktivierte Zeitpläne bleiben deaktiviert.
- Laufende Sicherungs- oder Snapshot-Vorgänge werden vor Beginn der Wiederherstellung abgebrochen. Wird ein Vorgang nicht innerhalb von 5 Minuten abgebrochen, wird die Wiederherstellung fortgesetzt und eine Warnung im Wiederherstellungs-CR-Status protokolliert.

Bevor Sie beginnen

Stellen Sie sicher, dass die Gültigkeitsdauer des AWS-Sitzungstokens für alle länger dauernden s3-Wiederherstellungsvorgänge ausreichend ist. Wenn das Token während des Wiederherstellungsvorgangs abläuft, kann der Vorgang fehlschlagen.

- Weitere Informationen zum Prüfen des Ablaufs des aktuellen Sitzungstokens finden Sie in der "AWS API-Dokumentation".
- Weitere Informationen zu Anmeldeinformationen für AWS-Ressourcen finden Sie in der "AWS IAM-Dokumentation".

Verwenden Sie einen CR

Schritte

1. Erstellen Sie die benutzerdefinierte Ressourcendatei (CR) und benennen Sie sie `trident-protect-snapshot-ipr-cr.yaml`.
2. Konfigurieren Sie in der von Ihnen erstellten Datei die folgenden Attribute:
   - **metadata.name**: (*Erforderlich*) Der Name dieser benutzerdefinierten Ressource; wählen Sie einen eindeutigen und sinnvollen Namen für Ihre Umgebung.
   - **spec.appVaultRef**: (*Erforderlich*) Der Name des AppVault, in dem die Snapshot-Inhalte gespeichert sind.
   - **spec.appArchivePath**: Der Pfad innerhalb von AppVault, in dem die Snapshot-Inhalte gespeichert sind. Sie können den folgenden Befehl verwenden, um diesen Pfad zu finden:

     ```
     kubectl get snapshots <SNAPSHOT_NAME> -n my-app-namespace -o jsonpath='{.status.appArchivePath}'
     ```

     ```
     ---
     apiVersion: protect.trident.netapp.io/v1
     kind: SnapshotInplaceRestore
     metadata:
       name: my-cr-name
       namespace: my-app-namespace
     spec:
       appVaultRef: appvault-name
       appArchivePath: my-snapshot-path
     ```

3. (*Optional*) Falls Sie nur bestimmte Ressourcen der Anwendung für die Wiederherstellung auswählen möchten, fügen Sie Filter hinzu, die Ressourcen mit bestimmten Bezeichnungen ein- oder ausschließen:

   HINWEIS Trident Protect wählt bestimmte Ressourcen automatisch aus, weil sie mit den von Ihnen ausgewählten Ressourcen in Beziehung stehen. Wenn Sie beispielsweise eine Ressource für einen persistenten Volume-Claim auswählen und diese einen zugehörigen Pod hat, wird Trident Protect auch den zugehörigen Pod wiederherstellen.

   - **resourceFilter.resourceSelectionCriteria**: (Für die Filterung erforderlich) Verwenden Sie `Include` oder `Exclude`, um eine in resourceMatchers definierte Ressource ein- oder auszuschließen. Fügen Sie die folgenden resourceMatchers-Parameter hinzu, um die Ressourcen zu definieren, die ein- oder auszuschließen sind:
     - **resourceFilter.resourceMatchers**: Ein Array von resourceMatcher-Objekten. Wenn Sie mehrere Elemente in diesem Array definieren, werden diese mit einer ODER-Verknüpfung verglichen und die Felder innerhalb jedes Elements (group, kind, version) werden mit einer UND-Verknüpfung verglichen.
       - **resourceMatchers[].group**: (*Optional*) Gruppe der zu filternden Ressource.

- **resourceMatchers[].kind**: (*Optional*) Art der zu filternden Ressource.
- **resourceMatchers[].version**: (*Optional*) Version der zu filternden Ressource.
- **resourceMatchers[].names**: (*Optional*) Namen im Kubernetes metadata.name-Feld der Ressource, die gefiltert werden soll.
- **resourceMatchers[].namespaces**: (*Optional*) Namespaces im Kubernetes metadata.name-Feld der Ressource, die gefiltert werden soll.
- **resourceMatchers[].labelSelectors**: (*Optional*) Label-Selektorzeichenfolge im Kubernetes-Metadatenfeld name der Ressource, wie definiert in der "Kubernetes-Dokumentation". Zum Beispiel: `"trident.netapp.io/os=linux"`.

  Beispiel:

```
spec:
  resourceFilter:
    resourceSelectionCriteria: "Include"
    resourceMatchers:
      - group: my-resource-group-1
        kind: my-resource-kind-1
        version: my-resource-version-1
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
      - group: my-resource-group-2
        kind: my-resource-kind-2
        version: my-resource-version-2
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
```

4. Nachdem Sie die `trident-protect-snapshot-ipr-cr.yaml` Datei mit den korrekten Werten gefüllt haben, wenden Sie die CR an:

```
kubectl apply -f trident-protect-snapshot-ipr-cr.yaml
```

Verwenden Sie die Befehlszeile

Schritte

1. Stellen Sie den Snapshot im ursprünglichen Namespace wieder her, indem Sie die Werte in Klammern durch Informationen aus Ihrer Umgebung ersetzen. Beispiel:

```
tridentctl-protect create snapshotinplacerestore <my_restore_name> \
--snapshot <namespace/snapshot_to_restore> \
-n <application_namespace>
```

Überprüfen Sie den Status eines Wiederherstellungsvorgangs

Sie können die Befehlszeile verwenden, um den Status eines Wiederherstellungsvorgangs zu überprüfen, der gerade läuft, abgeschlossen ist oder fehlgeschlagen ist.

Schritte

1. Verwenden Sie den folgenden Befehl, um den Status des Wiederherstellungsvorgangs abzurufen, und ersetzen Sie die Werte in Klammern durch Informationen aus Ihrer Umgebung:

```
kubectl get backuprestore -n <namespace_name> <my_restore_cr_name> -o
jsonpath='{.status}'
```

Verwenden Sie die erweiterten Trident Protect-Wiederherstellungseinstellungen

Sie können Wiederherstellungsvorgänge mithilfe erweiterter Einstellungen wie Annotationen, Namespace-Einstellungen und Speicheroptionen an Ihre spezifischen Anforderungen anpassen.

Namespace-Annotationen und -Labels während Wiederherstellungs- und Failover-Operationen

Während Wiederherstellungs- und Failover-Vorgängen werden die Labels und Annotationen im Ziel-Namensraum so angepasst, dass sie den Labels und Annotationen im Quell-Namensraum entsprechen. Labels oder Annotationen aus dem Quell-Namensraum, die im Ziel-Namensraum nicht existieren, werden hinzugefügt, und alle Labels oder Annotationen, die bereits vorhanden sind, werden überschrieben, um dem Wert aus dem Quell-Namensraum zu entsprechen. Labels oder Annotationen, die nur im Ziel-Namensraum existieren, bleiben unverändert.

HINWEIS	Wenn Sie Red Hat OpenShift verwenden, ist es wichtig, die entscheidende Rolle von Namespace-Annotationen in OpenShift-Umgebungen zu beachten. Namespace-Annotationen stellen sicher, dass wiederhergestellte Pods die entsprechenden Berechtigungen und Sicherheitskonfigurationen einhalten, die durch OpenShift Security Context Constraints (SCCs) definiert sind, und ohne Berechtigungsprobleme auf Volumes zugreifen können. Weitere Informationen finden Sie unter "OpenShift security context constraints Dokumentation".

Sie können verhindern, dass bestimmte Annotationen im Ziel-Namespace überschrieben werden, indem Sie die Kubernetes-Umgebungsvariable `RESTORE_SKIP_NAMESPACE_ANNOTATIONS` festlegen, bevor Sie die Wiederherstellungs- oder Failover-Operation durchführen. Beispiel:

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect \
  --set-string
restoreSkipNamespaceAnnotations="{<annotation_key_to_skip_1>,<annotation_k
ey_to_skip_2>}" \
  --reuse-values
```

HINWEIS	Bei der Durchführung einer Wiederherstellungs- oder Failover-Operation werden alle in `restoreSkipNamespaceAnnotations` und `restoreSkipNamespaceLabels` angegebenen Namespace-Annotationen und -Labels von der Wiederherstellungs- oder Failover-Operation ausgeschlossen. Stellen Sie sicher, dass diese Einstellungen während der initialen Helm-Installation konfiguriert sind. Weitere Informationen finden Sie unter "Konfigurieren Sie zusätzliche Trident Protect Helm-Chart-Einstellungen".

Wenn Sie die Quellanwendung mit Helm mit dem `--create-namespace` Flag installiert haben, wird dem `name` Label-Schlüssel eine besondere Behandlung zuteil. Während des Wiederherstellungs- oder Failover-Prozesses kopiert Trident Protect dieses Label in den Ziel-Namespace, aktualisiert jedoch den Wert auf den Wert des Ziel-Namespaces, wenn der Wert aus der Quelle mit dem Quell-Namespace übereinstimmt. Wenn dieser Wert nicht mit dem Quell-Namespace übereinstimmt, wird er unverändert in den Ziel-Namespace kopiert.

Beispiel

Das folgende Beispiel zeigt einen Quell- und einen Ziel-Namensraum mit jeweils unterschiedlichen Annotationen und Labels. Sie können den Zustand des Ziel-Namensraums vor und nach der Operation sehen und erkennen, wie die Annotationen und Labels im Ziel-Namensraum kombiniert oder überschrieben werden.

Vor dem Wiederherstellungs- oder Failover-Vorgang

Die folgende Tabelle veranschaulicht den Zustand der Beispiel-Quell- und Ziel-Namespaces vor der Wiederherstellungs- oder Failover-Operation:

Namensraum	Anmerkungen	Etiketten
Namespace ns-1 (Quelle)	• annotation.one/key: "updatedvalue" • annotation.two/key: "true"	• environment=production • compliance=hipaa • name=ns-1
Namespace ns-2 (Ziel)	• annotation.one/key: "true" • annotation.three/key: "false"	• role=database

Nach dem Wiederherstellungsvorgang

Die folgende Tabelle veranschaulicht den Zustand des Beispiel-Ziel-Namespace nach der Wiederherstellung oder dem Failover. Einige Schlüssel wurden hinzugefügt, einige wurden überschrieben, und das `name` Label wurde aktualisiert, um dem Ziel-Namespace zu entsprechen:

Namensraum	Anmerkungen	Etiketten
Namespace ns-2 (Ziel)	• annotation.one/key: "updatedvalue" • annotation.two/key: "true" • annotation.three/key: "false"	• name=ns-2 • compliance=hipaa • environment=production • role=database

Unterstützte Felder

In diesem Abschnitt werden zusätzliche Felder beschrieben, die für Wiederherstellungsvorgänge zur Verfügung stehen.

Speicherklassenzuordnung

Das `spec.storageClassMapping` Attribut definiert eine Zuordnung von einer Speicherklasse in der Quellanwendung zu einer neuen Speicherklasse im Zielcluster. Sie können dies verwenden, wenn Sie Anwendungen zwischen Clustern mit unterschiedlichen Speicherklassen migrieren oder das Speicher-Backend für BackupRestore-Operationen ändern.

Beispiel:

```
storageClassMapping:
  - destination: "destinationStorageClass1"
    source: "sourceStorageClass1"
  - destination: "destinationStorageClass2"
    source: "sourceStorageClass2"
```

Unterstützte Annotationen

Dieser Abschnitt listet die unterstützten Annotationen zur Konfiguration verschiedener Verhaltensweisen im System auf. Wenn eine Annotation nicht explizit vom Benutzer festgelegt wird, verwendet das System den Standardwert.

Anmerkung	Typ	Beschreibung	Standardwert
protect.trident.netapp.io/data-mover-timeout-sec	Zeichenkette	Die maximal zulässige Zeit (in Sekunden), in der der Datenübertragungsvorgang angehalten werden darf.	"300"
protect.trident.netapp.io/kopia-content-cache-size-limit-mb	Zeichenkette	Die maximale Größenbeschränkung (in Megabytes) für den Kopia-Inhaltscache.	"1000"
protect.trident.netapp.io/pvc-bind-timeout-sec	Zeichenkette	Maximale Zeit (in Sekunden), die auf neu erstellte PersistentVolumeClaims (PVCs) gewartet wird, um die `Bound` Phase zu erreichen, bevor der Vorgang fehlschlägt. Gilt für alle Restore-CR-Typen (BackupRestore, BackupInplaceRestore, SnapshotRestore, SnapshotInplaceRestore). Verwenden Sie einen höheren Wert, wenn Ihr Storage-Backend oder Cluster häufig mehr Zeit benötigt.	"1200" (20 Minuten)

Anwendungen mit NetApp SnapMirror und Trident Protect replizieren

Mit Trident Protect können Sie die asynchronen Replizierungsfunktionen der NetApp SnapMirror-Technologie nutzen, um Daten und Anwendungsänderungen von einem

Storage-Backend auf ein anderes zu replizieren, entweder im selben Cluster oder zwischen verschiedenen Clustern.

Namespace-Annotationen und -Labels während Wiederherstellungs- und Failover-Operationen

Während Wiederherstellungs- und Failover-Vorgängen werden die Labels und Annotationen im Ziel-Namensraum so angepasst, dass sie den Labels und Annotationen im Quell-Namensraum entsprechen. Labels oder Annotationen aus dem Quell-Namensraum, die im Ziel-Namensraum nicht existieren, werden hinzugefügt, und alle Labels oder Annotationen, die bereits vorhanden sind, werden überschrieben, um dem Wert aus dem Quell-Namensraum zu entsprechen. Labels oder Annotationen, die nur im Ziel-Namensraum existieren, bleiben unverändert.

HINWEIS Wenn Sie Red Hat OpenShift verwenden, ist es wichtig, die entscheidende Rolle von Namespace-Annotationen in OpenShift-Umgebungen zu beachten. Namespace-Annotationen stellen sicher, dass wiederhergestellte Pods die entsprechenden Berechtigungen und Sicherheitskonfigurationen einhalten, die durch OpenShift Security Context Constraints (SCCs) definiert sind, und ohne Berechtigungsprobleme auf Volumes zugreifen können. Weitere Informationen finden Sie unter "OpenShift security context constraints Dokumentation".

Sie können verhindern, dass bestimmte Annotationen im Ziel-Namespace überschrieben werden, indem Sie die Kubernetes-Umgebungsvariable `RESTORE_SKIP_NAMESPACE_ANNOTATIONS` festlegen, bevor Sie die Wiederherstellungs- oder Failover-Operation durchführen. Beispiel:

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect \
  --set-string
restoreSkipNamespaceAnnotations="{<annotation_key_to_skip_1>,<annotation_k
ey_to_skip_2>}" \
  --reuse-values
```

HINWEIS Bei der Durchführung einer Wiederherstellungs- oder Failover-Operation werden alle in `restoreSkipNamespaceAnnotations` und `restoreSkipNamespaceLabels` angegebenen Namespace-Annotationen und -Labels von der Wiederherstellungs- oder Failover-Operation ausgeschlossen. Stellen Sie sicher, dass diese Einstellungen während der initialen Helm-Installation konfiguriert sind. Weitere Informationen finden Sie unter "Konfigurieren Sie zusätzliche Trident Protect Helm-Chart-Einstellungen".

Wenn Sie die Quellanwendung mit Helm mit dem `--create-namespace` Flag installiert haben, wird dem `name` Label-Schlüssel eine besondere Behandlung zuteil. Während des Wiederherstellungs- oder Failover-Prozesses kopiert Trident Protect dieses Label in den Ziel-Namespace, aktualisiert jedoch den Wert auf den Wert des Ziel-Namespaces, wenn der Wert aus der Quelle mit dem Quell-Namespace übereinstimmt. Wenn dieser Wert nicht mit dem Quell-Namespace übereinstimmt, wird er unverändert in den Ziel-Namespace kopiert.

Beispiel

Das folgende Beispiel zeigt einen Quell- und einen Ziel-Namensraum mit jeweils unterschiedlichen Annotationen und Labels. Sie können den Zustand des Ziel-Namensraums vor und nach der Operation sehen und erkennen, wie die Annotationen und Labels im Ziel-Namensraum kombiniert oder überschrieben werden.

Vor dem Wiederherstellungs- oder Failover-Vorgang

Die folgende Tabelle veranschaulicht den Zustand der Beispiel-Quell- und Ziel-Namespaces vor der Wiederherstellungs- oder Failover-Operation:

Namensraum	Anmerkungen	Etiketten
Namespace ns-1 (Quelle)	• annotation.one/key: "updatedvalue" • annotation.two/key: "true"	• environment=production • compliance=hipaa • name=ns-1
Namespace ns-2 (Ziel)	• annotation.one/key: "true" • annotation.three/key: "false"	• role=database

Nach dem Wiederherstellungsvorgang

Die folgende Tabelle veranschaulicht den Zustand des Beispiel-Ziel-Namespace nach der Wiederherstellung oder dem Failover. Einige Schlüssel wurden hinzugefügt, einige wurden überschrieben, und das `name` Label wurde aktualisiert, um dem Ziel-Namespace zu entsprechen:

Namensraum	Anmerkungen	Etiketten
Namespace ns-2 (Ziel)	• annotation.one/key: "updatedvalue" • annotation.two/key: "true" • annotation.three/key: "false"	• name=ns-2 • compliance=hipaa • environment=production • role=database

HINWEIS	Sie können Trident Protect so konfigurieren, dass Dateisysteme während Datensicherungsoperationen eingefroren und wieder freigegeben werden. "Erfahren Sie mehr über die Konfiguration des Einfrierens von Dateisystemen mit Trident Protect".

Ausführungshooks während Failover- und Reverse-Operationen

Wenn Sie eine AppMirror-Beziehung zum Schutz Ihrer Anwendung verwenden, gibt es bestimmte Verhaltensweisen im Zusammenhang mit Ausführungs-Hooks, die Sie während Failover- und Rückwärtsoperationen beachten sollten.

- Während des Failovers werden die Ausführungshooks automatisch vom Quell-Cluster auf den Ziel-Cluster kopiert. Sie müssen sie nicht manuell neu erstellen. Nach dem Failover sind die Ausführungshooks in der Anwendung vorhanden und werden bei allen relevanten Aktionen ausgeführt.
- Während der Reverse- oder Reverse-Resync werden alle vorhandenen Ausführungs-Hooks der Anwendung entfernt. Wenn die Quellanwendung zur Zielanwendung wird, sind diese Ausführungs-Hooks nicht mehr gültig und werden gelöscht, um ihre Ausführung zu verhindern.

Weitere Informationen zu Execution Hooks finden Sie unter "Trident Protect-Ausführungshooks verwalten".

Eine Replikationsbeziehung einrichten

Die Einrichtung einer Replikationsbeziehung umfasst Folgendes:

- Auswahl, wie häufig Trident Protect einen App-Snapshot erstellen soll (der sowohl die Kubernetes-Ressourcen der App als auch die Volume-Snapshots für jedes der App-Volumes umfasst)
- Auswahl des Replikationszeitplans (einschließlich Kubernetes-Ressourcen sowie persistenter Volume-Daten)
- Einstellen des Zeitpunkts für die Aufnahme des Snapshots

Schritte

1. Erstellen Sie auf dem Quell-Cluster ein AppVault für die Quellanwendung. Passen Sie je nach Speicheranbieter ein Beispiel in "AppVault benutzerdefinierte Ressourcen" an Ihre Umgebung an:

Erstellen Sie ein AppVault mit einem CR

a. Erstellen Sie die benutzerdefinierte Ressourcendatei (CR) und benennen Sie sie (zum Beispiel `trident-protect-appvault-primary-source.yaml`).

b. Konfigurieren Sie die folgenden Attribute:

- **metadata.name**: (*Erforderlich*) Der Name der AppVault Custom Resource. Notieren Sie sich den Namen, den Sie wählen, da andere für eine Replikationsbeziehung benötigte CR-Dateien auf diesen Wert verweisen.
- **spec.providerConfig**: (*Erforderlich*) Speichert die Konfiguration, die für den Zugriff auf AppVault mit dem angegebenen Provider erforderlich ist. Wählen Sie einen bucketName und alle weiteren erforderlichen Details für Ihren Provider. Notieren Sie sich die Werte, die Sie wählen, da andere für eine Replikationsbeziehung benötigte CR-Dateien auf diese Werte verweisen. Siehe "AppVault benutzerdefinierte Ressourcen" für Beispiele von AppVault CRs mit anderen Providern.
- **spec.providerCredentials**: (*Erforderlich*) Speichert Verweise auf alle Anmeldeinformationen, die für den Zugriff auf den AppVault mit dem angegebenen Anbieter erforderlich sind.
  - **spec.providerCredentials.valueFromSecret**: (*Erforderlich*) Gibt an, dass der Anmeldeinformationswert aus einem Geheimnis stammen sollte.
    - **Schlüssel**: (*Erforderlich*) Der gültige Schlüssel des Geheimnisses, der ausgewählt werden soll.
    - **Name**: (*Erforderlich*) Name des Geheimnisses, das den Wert für dieses Feld enthält. Muss sich im selben Namespace befinden.
  - **spec.providerCredentials.secretAccessKey**: (*Erforderlich*) Der Zugriffsschlüssel, der für den Zugriff auf den Provider verwendet wird. Der **name** muss mit **spec.providerCredentials.valueFromSecret.name** übereinstimmen.
- **spec.providerType**: (*Erforderlich*) Legt fest, was die Datensicherung bereitstellt; beispielsweise NetApp ONTAP S3, generisches S3, Google Cloud oder Microsoft Azure. Mögliche Werte:
  - aws
  - azure
  - gcp
  - generic-s3
  - ontap-s3
  - storagegrid-s3

c. Nachdem Sie die `trident-protect-appvault-primary-source.yaml` Datei mit den korrekten Werten gefüllt haben, wenden Sie die CR an:

```
kubectl apply -f trident-protect-appvault-primary-source.yaml -n
trident-protect
```

Erstellen Sie ein AppVault mit der CLI

a. Erstellen Sie die AppVault, indem Sie die Werte in Klammern durch Informationen aus Ihrer Umgebung ersetzen:

```
tridentctl-protect create vault Azure <vault-name> --account
<account-name> --bucket <bucket-name> --secret <secret-name> -n
trident-protect
```

2. Erstellen Sie auf dem Quell-Cluster die Quellanwendungs-CR:

Erstellen Sie die Quellanwendung mithilfe eines CR

a. Erstellen Sie die benutzerdefinierte Ressourcendatei (CR) und benennen Sie sie (zum Beispiel `trident-protect-app-source.yaml`).

b. Konfigurieren Sie die folgenden Attribute:

- **metadata.name**: (*Erforderlich*) Der Name der benutzerdefinierten Anwendungsressource. Notieren Sie sich den Namen, den Sie wählen, da andere für eine Replikationsbeziehung benötigte CR-Dateien auf diesen Wert verweisen.
- **spec.includedNamespaces**: (*Erforderlich*) Ein Array von Namespaces und zugehörigen Labels. Verwenden Sie Namespace-Namen und grenzen Sie optional den Geltungsbereich der Namespaces mit Labels ein, um Ressourcen anzugeben, die in den hier aufgeführten Namespaces vorhanden sind. Der Anwendungsnamespace muss Teil dieses Arrays sein.

  Beispiel YAML:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: Application
metadata:
  name: my-app-name
  namespace: my-app-namespace
spec:
  includedNamespaces:
    - namespace: my-app-namespace
      labelSelector: {}
```

c. Nachdem Sie die `trident-protect-app-source.yaml` Datei mit den korrekten Werten gefüllt haben, wenden Sie die CR an:

```
kubectl apply -f trident-protect-app-source.yaml -n my-app-
namespace
```

Erstellen Sie die Quellanwendung mithilfe der CLI

a. Erstellen Sie die Quellanwendung. Beispiel:

```
tridentctl-protect create app <my-app-name> --namespaces
<namespaces-to-be-included> -n <my-app-namespace>
```

3. Optional können Sie auf dem Quell-Cluster einen Snapshot der Quellanwendung erstellen. Dieser Snapshot dient als Grundlage für die Anwendung auf dem Ziel-Cluster. Wenn Sie diesen Schritt überspringen, müssen Sie warten, bis der nächste geplante Snapshot ausgeführt wird, damit Sie einen aktuellen Snapshot haben. Um einen bedarfsgesteuerten Snapshot zu erstellen, siehe "Erstellen Sie einen On-Demand-Snapshot".

4. Erstellen Sie auf dem Quell-Cluster den Replikationszeitplan CR:

HINWEIS

Zusätzlich zum unten angegebenen Zeitplan wird empfohlen, einen separaten Zeitplan für tägliche Snapshots mit einer Aufbewahrungsdauer von 7 Tagen zu erstellen, um einen gemeinsamen Snapshot zwischen verbundenen ONTAP Clustern zu gewährleisten. Dadurch ist sichergestellt, dass Snapshots bis zu 7 Tage lang verfügbar sind, aber die Aufbewahrungsdauer kann basierend auf den Benutzeranforderungen angepasst werden.

Im Falle eines Failovers kann das System diese Snapshots bis zu 7 Tage lang für Rückoperationen nutzen. Dieser Ansatz macht den Rückprozess schneller und effizienter, da nur die seit dem letzten Snapshot vorgenommenen Änderungen übertragen werden, nicht alle Daten.

Wenn ein bestehender Zeitplan für die Anwendung bereits die gewünschten Aufbewahrungsanforderungen erfüllt, sind keine zusätzlichen Zeitpläne erforderlich.

Erstellen Sie den Replikationszeitplan mithilfe einer CR

a. Erstellen Sie einen Replikationszeitplan für die Quellanwendung:

   i. Erstellen Sie die benutzerdefinierte Ressourcendatei (CR) und benennen Sie sie (zum Beispiel `trident-protect-schedule.yaml`).

   ii. Konfigurieren Sie die folgenden Attribute:

   - **metadata.name**: (*Erforderlich*) Der Name der benutzerdefinierten Zeitplanressource.
   - **spec.appVaultRef**: (*Erforderlich*) Dieser Wert muss mit dem Feld metadata.name der AppVault für die Quellanwendung übereinstimmen.
   - **spec.applicationRef**: (*Erforderlich*) Dieser Wert muss mit dem Feld metadata.name der Quellanwendung CR übereinstimmen.
   - **spec.backupRetention**: (*Erforderlich*) Dieses Feld ist erforderlich und der Wert muss auf 0 gesetzt werden.
   - **spec.enabled**: Muss auf true gesetzt werden.
   - **spec.granularity**: Muss auf `Custom` gesetzt werden.
   - **spec.recurrenceRule**: Definieren Sie ein Startdatum in UTC-Zeit und ein Wiederholungsintervall.
   - **spec.snapshotRetention**: Muss auf 2 eingestellt sein.

     Beispiel YAML:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: Schedule
metadata:
  name: appmirror-schedule
  namespace: my-app-namespace
spec:
  appVaultRef: my-appvault-name
  applicationRef: my-app-name
  backupRetention: "0"
  enabled: true
  granularity: Custom
  recurrenceRule: |-
    DTSTART:20220101T000200Z
    RRULE:FREQ=MINUTELY;INTERVAL=5
  snapshotRetention: "2"
```

   i. Nachdem Sie die `trident-protect-schedule.yaml` Datei mit den korrekten Werten gefüllt haben, wenden Sie die CR an:

```
kubectl apply -f trident-protect-schedule.yaml -n my-app-
namespace
```

Erstellen Sie den Replikationszeitplan mithilfe der CLI

a. Erstellen Sie den Replikationszeitplan, und ersetzen Sie die Werte in Klammern durch Informationen aus Ihrer Umgebung:

```
tridentctl-protect create schedule --name appmirror-schedule
--app <my_app_name> --appvault <my_app_vault> --granularity
Custom --recurrence-rule <rule> --snapshot-retention
<snapshot_retention_count> -n <my_app_namespace>
```

Beispiel:

```
tridentctl-protect create schedule --name appmirror-schedule
--app <my_app_name> --appvault <my_app_vault> --granularity
Custom --recurrence-rule  "DTSTART:20220101T000200Z
\nRRULE:FREQ=MINUTELY;INTERVAL=5" --snapshot-retention 2 -n
<my_app_namespace>
```

5. Erstellen Sie auf dem Ziel-Cluster eine Quellanwendungs-AppVault-CR, die mit der AppVault-CR identisch ist, die Sie auf dem Quell-Cluster angewendet haben, und benennen Sie sie (zum Beispiel `trident-protect-appvault-primary-destination.yaml`).
6. Wenden Sie die CR an:

```
kubectl apply -f trident-protect-appvault-primary-destination.yaml -n
trident-protect
```

7. Erstellen Sie eine Ziel-AppVault CR für die Zielanwendung im Ziel-Cluster. Passen Sie je nach Speicheranbieter ein Beispiel in "AppVault benutzerdefinierte Ressourcen" an Ihre Umgebung an:
   a. Erstellen Sie die benutzerdefinierte Ressourcendatei (CR) und benennen Sie sie (zum Beispiel `trident-protect-appvault-secondary-destination.yaml`).
   b. Konfigurieren Sie die folgenden Attribute:
      - **metadata.name**: (*Erforderlich*) Der Name der AppVault Custom Resource. Notieren Sie sich den Namen, den Sie wählen, da andere für eine Replikationsbeziehung benötigte CR-Dateien auf diesen Wert verweisen.
      - **spec.providerConfig**: (*Erforderlich*) Speichert die Konfiguration, die für den Zugriff auf AppVault mit dem angegebenen Provider erforderlich ist. Wählen Sie einen `bucketName` und alle weiteren erforderlichen Details für Ihren Provider. Notieren Sie sich die Werte, die Sie wählen, da andere für eine Replikationsbeziehung benötigte CR-Dateien auf diese Werte verweisen. Siehe "AppVault benutzerdefinierte Ressourcen" für Beispiele von AppVault CRs mit anderen Providern.

- **spec.providerCredentials**: (*Erforderlich*) Speichert Verweise auf alle Anmeldeinformationen, die für den Zugriff auf den AppVault mit dem angegebenen Anbieter erforderlich sind.
  - **spec.providerCredentials.valueFromSecret**: (*Erforderlich*) Gibt an, dass der Anmeldeinformationswert aus einem Geheimnis stammen sollte.
    - **Schlüssel**: (*Erforderlich*) Der gültige Schlüssel des Geheimnisses, der ausgewählt werden soll.
    - **Name**: (*Erforderlich*) Name des Geheimnisses, das den Wert für dieses Feld enthält. Muss sich im selben Namespace befinden.
  - **spec.providerCredentials.secretAccessKey**: (*Erforderlich*) Der Zugriffsschlüssel, der für den Zugriff auf den Provider verwendet wird. Der **name** muss mit **spec.providerCredentials.valueFromSecret.name** übereinstimmen.
- **spec.providerType**: (*Erforderlich*) Legt fest, was die Datensicherung bereitstellt; beispielsweise NetApp ONTAP S3, generisches S3, Google Cloud oder Microsoft Azure. Mögliche Werte:
  - aws
  - azure
  - gcp
  - generic-s3
  - ontap-s3
  - storagegrid-s3

c. Nachdem Sie die `trident-protect-appvault-secondary-destination.yaml` Datei mit den korrekten Werten gefüllt haben, wenden Sie die CR an:

```
kubectl apply -f trident-protect-appvault-secondary-destination.yaml
-n trident-protect
```

8. Erstellen Sie auf dem Ziel-Cluster eine AppMirrorRelationship CR-Datei.

HINWEIS	Bei Verwendung einer CR muss der Ziel-Namespace vor der Anwendung der CR manuell erstellt werden. Trident Protect erstellt Namespaces automatisch nur bei Verwendung der CLI.

Erstellen Sie eine AppMirrorRelationship mit einem CR

a. Erstellen Sie die benutzerdefinierte Ressourcendatei (CR) und benennen Sie sie (zum Beispiel `trident-protect-relationship.yaml`).

b. Konfigurieren Sie die folgenden Attribute:

- **metadata.name:** (Erforderlich) Der Name der AppMirrorRelationship benutzerdefinierten Ressource.
- **spec.destinationAppVaultRef**: (*Erforderlich*) Dieser Wert muss mit dem Namen der AppVault für die Zielanwendung auf dem Ziel-Cluster übereinstimmen.
- **spec.namespaceMapping**: (*Erforderlich*) Die Ziel- und Quell-Namespaces müssen mit dem in der jeweiligen Anwendungs-CR definierten Anwendungs-Namespace übereinstimmen.
- **spec.sourceAppVaultRef**: (*Erforderlich*) Dieser Wert muss mit dem Namen der AppVault für die Quellanwendung übereinstimmen.
- **spec.sourceApplicationName**: (*Erforderlich*) Dieser Wert muss mit dem Namen der Quellanwendung übereinstimmen, die Sie im Quellanwendungs-CR definiert haben.
- **spec.sourceApplicationUID**: (Erforderlich) Dieser Wert muss mit der UID der Quellanwendung übereinstimmen, die Sie im Quellanwendungs-CR definiert haben.
- **spec.storageClassName**: (*Optional*) Wählen Sie den Namen einer gültigen Speicherklasse auf dem Cluster. Die Speicherklasse muss mit einer ONTAP Storage-VM verknüpft sein, die mit der Quellumgebung gepaart ist. Wenn keine Speicherklasse angegeben wird, wird standardmäßig die Standard-Speicherklasse auf dem Cluster verwendet.
- **spec.recurrenceRule**: Definieren Sie ein Startdatum in UTC-Zeit und ein Wiederholungsintervall.

  Beispiel YAML:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: AppMirrorRelationship
metadata:
  name: amr-16061e80-1b05-4e80-9d26-d326dc1953d8
  namespace: my-app-namespace
spec:
  desiredState: Established
  destinationAppVaultRef: generic-s3-trident-protect-dst-bucket-
8fe0b902-f369-4317-93d1-ad7f2edc02b5
  namespaceMapping:
    - destination: my-app-namespace
      source: my-app-namespace
  recurrenceRule: |-
    DTSTART:20220101T000200Z
    RRULE:FREQ=MINUTELY;INTERVAL=5
  sourceAppVaultRef: generic-s3-trident-protect-src-bucket-
b643cc50-0429-4ad5-971f-ac4a83621922
  sourceApplicationName: my-app-name
  sourceApplicationUID: 7498d32c-328e-4ddd-9029-122540866aeb
  storageClassName: sc-vsim-2
```

c. Nachdem Sie die `trident-protect-relationship.yaml` Datei mit den korrekten Werten gefüllt haben, wenden Sie die CR an:

```
kubectl apply -f trident-protect-relationship.yaml -n my-app-
namespace
```

Erstellen Sie eine AppMirrorRelationship mithilfe der CLI

a. Erstellen und wenden Sie das AppMirrorRelationship-Objekt an, indem Sie die Werte in Klammern durch Informationen aus Ihrer Umgebung ersetzen:

```
tridentctl-protect create appmirrorrelationship
<name_of_appmirorrelationship> --destination-app-vault
<my_vault_name> --source-app-vault <my_vault_name> --recurrence
-rule <rule> --namespace-mapping <ns_mapping> --source-app-id
<source_app_UID> --source-app <my_source_app_name> --storage
-class <storage_class_name> -n <application_namespace>
```

Beispiel:

```
tridentctl-protect create appmirrorrelationship my-amr
--destination-app-vault appvault2 --source-app-vault appvault1
--recurrence-rule
"DTSTART:20220101T000200Z\nRRULE:FREQ=MINUTELY;INTERVAL=5"
--source-app my-app --namespace-mapping "my-source-ns1:my-dest-
ns1,my-source-ns2:my-dest-ns2" --source-app-id 373f24c1-5769-
404c-93c3-5538af6ccc36 --storage-class my-storage-class -n my-
dest-ns1
```

9. (*Optional*) Überprüfen Sie auf dem Ziel-Cluster den Status und den Zustand der Replikationsbeziehung:

```
kubectl get amr -n my-app-namespace <relationship name> -o=jsonpath
='{.status}' | jq
```

Failover zum Ziel-Cluster

Mit Trident Protect können Sie replizierte Anwendungen auf einen Ziel-Cluster umschalten. Dieses Verfahren stoppt die Replikationsbeziehung und bringt die Anwendung im Ziel-Cluster online. Trident Protect stoppt die Anwendung im Quell-Cluster nicht, wenn sie dort betriebsbereit war.

Schritte

1. Bearbeiten Sie auf dem Ziel-Cluster die AppMirrorRelationship CR-Datei (zum Beispiel `trident-protect-relationship.yaml`) und ändern Sie den Wert von **spec.desiredState** zu `Promoted`.
2. Speichern Sie die CR-Datei.
3. Wenden Sie die CR an:

```
kubectl apply -f trident-protect-relationship.yaml -n my-app-namespace
```

4. (*Optional*) Erstellen Sie alle benötigten Schutzzeitpläne für die übernommene Anwendung.
5. (*Optional*) Überprüfen Sie den Status und Zustand der Replikationsbeziehung:

```
kubectl get amr -n my-app-namespace <relationship name> -o=jsonpath
='{.status}' | jq
```

Eine fehlgeschlagene Replikationsbeziehung erneut synchronisieren

Der Resynchronisierungsvorgang stellt die Replikationsbeziehung wieder her. Nach der Durchführung eines Resynchronisierungsvorgangs wird die ursprüngliche Quellanwendung zur aktiven Anwendung und alle Änderungen, die an der aktiven Anwendung im Ziel-Cluster vorgenommen wurden, werden verworfen.

Der Prozess stoppt die App auf dem Ziel-Cluster, bevor die Replikation wiederhergestellt wird.

WICHTIG	Alle Daten, die während des Failovers in die Zielanwendung geschrieben werden, gehen verloren.

Schritte

1. Optional: Erstellen Sie auf dem Quell-Cluster einen Snapshot der Quellanwendung. Dadurch wird sichergestellt, dass die neuesten Änderungen vom Quell-Cluster erfasst werden.
2. Bearbeiten Sie auf dem Ziel-Cluster die AppMirrorRelationship CR-Datei (zum Beispiel `trident-protect-relationship.yaml`) und ändern Sie den Wert von spec.desiredState zu `Established`.
3. Speichern Sie die CR-Datei.
4. Wenden Sie die CR an:

```
kubectl apply -f trident-protect-relationship.yaml -n my-app-namespace
```

5. Wenn Sie auf dem Ziel-Cluster Schutzzeitpläne für die ausgefallene Anwendung erstellt haben, entfernen Sie diese. Alle verbleibenden Zeitpläne verursachen Fehler bei Volume-Snapshots.

Reverse resync einer fehlgeschlagenen Replikationsbeziehung

Wenn Sie eine fehlgeschlagene Replikationsbeziehung rücksynchronisieren, wird die Zielanwendung zur Quellanwendung und die Quellanwendung zur Zielanwendung. Änderungen, die während des Failovers an der Zielanwendung vorgenommen wurden, bleiben erhalten.

Schritte

1. Löschen Sie auf dem ursprünglichen Ziel-Cluster die AppMirrorRelationship CR. Dadurch wird das Ziel-Cluster zum Quell-Cluster. Wenn auf dem neuen Ziel-Cluster noch Datensicherungsstrategien vorhanden sind, entfernen Sie diese.
2. Richten Sie eine Replikationsbeziehung ein, indem Sie die CR-Dateien, die Sie ursprünglich zum Einrichten der Beziehung verwendet haben, auf die gegenüberliegenden Cluster anwenden.
3. Stellen Sie sicher, dass das neue Ziel (ursprünglicher Quell-Cluster) mit beiden AppVault CRs konfiguriert ist.
4. Richten Sie eine Replikationsbeziehung auf dem gegenüberliegenden Cluster ein, und konfigurieren Sie Werte für die umgekehrte Richtung.

Replikationsrichtung der Anwendung umkehren

Wenn Sie die Replikationsrichtung umkehren, verschiebt Trident Protect die Anwendung zum Ziel-Storage-Backend und repliziert gleichzeitig weiterhin zurück zum ursprünglichen Quell-Storage-Backend. Trident Protect stoppt die Quellanwendung und repliziert die Daten zum Ziel, bevor auf die Zielanwendung umgeschaltet wird.

In dieser Situation tauschen Sie den Quell-Cluster und den Ziel-Cluster.

Schritte

1. Erstellen Sie auf dem Quell-Cluster einen Shutdown-Snapshot:

Erstellen Sie einen Shutdown-Snapshot mithilfe eines CR

a. Deaktivieren Sie die Zeitpläne der Datensicherungsstrategie für die Quellanwendung.

b. Erstellen Sie eine ShutdownSnapshot CR-Datei:

   i. Erstellen Sie die benutzerdefinierte Ressourcendatei (CR) und benennen Sie sie (zum Beispiel `trident-protect-shutdownsnapshot.yaml`).

   ii. Konfigurieren Sie die folgenden Attribute:

   - **metadata.name**: (*Erforderlich*) Der Name der benutzerdefinierten Ressource.
   - **spec.AppVaultRef**: (*Erforderlich*) Dieser Wert muss mit dem Feld metadata.name der AppVault für die Quellanwendung übereinstimmen.
   - **spec.ApplicationRef**: (*Erforderlich*) Dieser Wert muss mit dem Feld metadata.name der Quellanwendungs-CR-Datei übereinstimmen.

     Beispiel YAML:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: ShutdownSnapshot
metadata:
  name: replication-shutdown-snapshot-afc4c564-e700-4b72-86c3-
c08a5dbe844e
  namespace: my-app-namespace
spec:
  appVaultRef: generic-s3-trident-protect-src-bucket-04b6b4ec-
46a3-420a-b351-45795e1b5e34
  applicationRef: my-app-name
```

c. Nachdem Sie die `trident-protect-shutdownsnapshot.yaml` Datei mit den korrekten Werten gefüllt haben, wenden Sie die CR an:

```
kubectl apply -f trident-protect-shutdownsnapshot.yaml -n my-app-
namespace
```

Erstellen Sie einen Shutdown-Snapshot mit der CLI

a. Erstellen Sie den Shutdown-Snapshot, indem Sie die Werte in Klammern durch Informationen aus Ihrer Umgebung ersetzen. Beispiel:

```
tridentctl-protect create shutdownsnapshot <my_shutdown_snapshot>
--appvault <my_vault> --app <app_to_snapshot> -n
<application_namespace>
```

2. Auf dem Quell-Cluster, nachdem der Shutdown-Snapshot abgeschlossen ist, rufen Sie den Status des Shutdown-Snapshots ab:

```
kubectl get shutdownsnapshot -n my-app-namespace
<shutdown_snapshot_name> -o yaml
```

3. Ermitteln Sie auf dem Quell-Cluster den Wert von **shutdownsnapshot.status.appArchivePath** mit dem folgenden Befehl und notieren Sie den letzten Teil des Dateipfads (auch Basisname genannt; dies ist alles nach dem letzten Schrägstrich):

```
k get shutdownsnapshot -n my-app-namespace <shutdown_snapshot_name> -o
jsonpath='{.status.appArchivePath}'
```

4. Führen Sie ein Failover vom neuen Ziel-Cluster zum neuen Quell-Cluster mit der folgenden Änderung durch:

HINWEIS	Fügen Sie in Schritt 2 des Failover-Verfahrens das `spec.promotedSnapshot` Feld in die AppMirrorRelationship CR-Datei ein und setzen Sie dessen Wert auf den Basisnamen, den Sie in Schritt 3 oben notiert haben.

5. Führen Sie die umgekehrten Resynchronisierungsschritte in Reverse resync einer fehlgeschlagenen Replikationsbeziehung aus.
6. Aktivieren Sie Schutzzeitpläne auf dem neuen Quell-Cluster.

Ergebnis

Die folgenden Aktionen erfolgen aufgrund der umgekehrten Replikation:

- Es wird eine Snapshot der Kubernetes-Ressourcen der ursprünglichen Quellanwendung erstellt.
- Die Pods der ursprünglichen Quell-App werden ordnungsgemäß gestoppt, indem die Kubernetes-Ressourcen der App gelöscht werden (PVCs und PVs bleiben erhalten).
- Nach dem Herunterfahren der Pods werden Snapshots der App-Volumes erstellt und repliziert.
- Die SnapMirror Beziehungen wurden aufgehoben, wodurch die Zielvolumes für lesen/schreiben bereit sind.
- Die Kubernetes-Ressourcen der App werden aus dem Snapshot vor dem Herunterfahren wiederhergestellt, wobei die Volume-Daten verwendet werden, die nach dem Herunterfahren der ursprünglichen Quell-App repliziert wurden.
- Die Replikation wird in umgekehrter Richtung wiederhergestellt.

Failback von Applikationen auf den ursprünglichen Quell-Cluster

Mit Trident Protect können Sie „Failback“ nach einem Failover-Vorgang durchführen, indem Sie die folgende Abfolge von Schritten ausführen. In diesem Workflow zur Wiederherstellung der ursprünglichen Replikationsrichtung repliziert (resynchronisiert) Trident Protect alle Anwendungsänderungen zurück in die ursprüngliche Quellanwendung, bevor die Replikationsrichtung umgekehrt wird.

Dieser Prozess beginnt mit einer Beziehung, die einen Failover auf ein Ziel abgeschlossen hat und umfasst die folgenden Schritte:

- Beginnen Sie mit einem übergewechselten Zustand.
- Reverse resync die Replikationsbeziehung.

ACHTUNG	Führen Sie keine normale Resynchronisierungsoperation durch, da dadurch Daten, die während des Failover-Vorgangs auf den Ziel-Cluster geschrieben wurden, verworfen werden.

- Die Replikationsrichtung umkehren.

Schritte

1. Führen Sie die Reverse resync einer fehlgeschlagenen Replikationsbeziehung Schritte aus.
2. Führen Sie die Replikationsrichtung der Anwendung umkehren Schritte aus.

Eine Replikationsbeziehung löschen

Sie können eine Replikationsbeziehung jederzeit löschen. Wenn Sie die Replikationsbeziehung der Anwendung löschen, entstehen zwei separate Anwendungen ohne Beziehung zueinander.

Schritte

1. Löschen Sie im aktuellen Ziel-Cluster die AppMirrorRelationship CR:

```
kubectl delete -f trident-protect-relationship.yaml -n my-app-namespace
```

Anwendungen mit Trident Protect migrieren

Sie können Ihre Anwendungen zwischen Clustern oder auf verschiedene Storage-Klassen migrieren, indem Sie Sicherungsdaten wiederherstellen.

HINWEIS	Bei der Migration einer Anwendung werden alle für die Anwendung konfigurierten Ausführungs-Hooks mit der Anwendung migriert. Wenn ein Ausführungs-Hook nach der Wiederherstellung vorhanden ist, wird er automatisch als Teil des Wiederherstellungsvorgangs ausgeführt.

Sicherungs- und Wiederherstellungsvorgänge

Um Sicherungs- und Wiederherstellungsvorgänge für die folgenden Szenarien durchzuführen, können Sie bestimmte Sicherungs- und Wiederherstellungsaufgaben automatisieren.

Klon auf denselben Cluster

Um eine Anwendung auf denselben Cluster zu klonen, erstellen Sie einen Snapshot oder eine Sicherung und stellen Sie die Daten auf denselben Cluster wieder her.

Schritte

1. Führen Sie einen der folgenden Schritte aus:
   a. "Erstellen Sie einen Snapshot".
   b. "Erstelle eine Sicherungskopie".

2. Führen Sie auf demselben Cluster einen der folgenden Schritte aus, je nachdem, ob Sie einen Snapshot oder ein Backup erstellt haben:
   a. "Stellen Sie Ihre Daten aus dem Snapshot wieder her".
   b. "Stellen Sie Ihre Daten aus der Sicherung wieder her".

Klonen auf anderen Cluster

Um eine Anwendung auf einen anderen Cluster zu klonen (clusterübergreifendes Klonen), erstellen Sie eine Sicherung auf dem Quell-Cluster und stellen Sie diese Sicherung anschließend auf dem Ziel-Cluster wieder her. Stellen Sie sicher, dass Trident Protect auf dem Ziel-Cluster installiert ist.

HINWEIS	Sie können eine Anwendung zwischen verschiedenen Clustern replizieren, indem Sie "SnapMirror-Replikation".

Schritte

1. "Erstelle eine Sicherungskopie".
2. Stellen Sie sicher, dass der AppVault CR für den Objektspeicher-Bucket, der die Sicherung enthält, auf dem Ziel-Cluster konfiguriert wurde.
3. Auf dem Ziel-Cluster "Stellen Sie Ihre Daten aus der Sicherung wieder her".

Anwendungen von einer Speicherklasse zu einer anderen Speicherklasse migrieren

Sie können Anwendungen von einer Speicherklasse in eine andere Speicherklasse migrieren, indem Sie ein Backup in der Ziel-Speicherklasse wiederherstellen.

Zum Beispiel (ohne die Geheimnisse aus dem Wiederherstellungs-CR):

```
apiVersion: protect.trident.netapp.io/v1
kind: SnapshotRestore
metadata:
  name: "${snapshotRestoreCRName}"
spec:
  appArchivePath: "${snapshotArchivePath}"
  appVaultRef: "${appVaultCRName}"
  namespaceMapping:
    - destination: "${destinationNamespace}"
      source: "${sourceNamespace}"
  storageClassMapping:
    - destination: "${destinationStorageClass}"
      source: "${sourceStorageClass}"
  resourceFilter:
    resourceMatchers:
      kind: Secret
      version: v1
    resourceSelectionCriteria: exclude
```

Stellen Sie den Snapshot mithilfe einer CR wieder her

Schritte

1. Erstellen Sie die benutzerdefinierte Ressourcendatei (CR) und benennen Sie sie `trident-protect-snapshot-restore-cr.yaml`.
2. Konfigurieren Sie in der von Ihnen erstellten Datei die folgenden Attribute:
   - **metadata.name**: (*Erforderlich*) Der Name dieser benutzerdefinierten Ressource; wählen Sie einen eindeutigen und sinnvollen Namen für Ihre Umgebung.
   - **spec.appArchivePath**: Der Pfad innerhalb von AppVault, in dem die Snapshot-Inhalte gespeichert sind. Sie können den folgenden Befehl verwenden, um diesen Pfad zu finden:

     ```
     kubectl get snapshots <my-snapshot-name> -n trident-protect -o jsonpath='{.status.appArchivePath}'
     ```

   - **spec.appVaultRef**: (*Erforderlich*) Der Name des AppVault, in dem die Snapshot-Inhalte gespeichert sind.
   - **spec.namespaceMapping**: Die Zuordnung des Quell-Namespace des Wiederherstellungsvorgangs zum Ziel-Namespace. Ersetzen Sie `my-source-namespace` und `my-destination-namespace` durch Informationen aus Ihrer Umgebung.

     ```
     ---
     apiVersion: protect.trident.netapp.io/v1
     kind: SnapshotRestore
     metadata:
       name: my-cr-name
       namespace: trident-protect
     spec:
       appArchivePath: my-snapshot-path
       appVaultRef: appvault-name
       namespaceMapping: [{"source": "my-source-namespace", "destination": "my-destination-namespace"}]
     ```

3. Optional können Sie, falls Sie nur bestimmte Ressourcen der Anwendung für die Wiederherstellung auswählen müssen, Filter hinzufügen, die Ressourcen mit bestimmten Bezeichnungen ein- oder ausschließen:
   - **resourceFilter.resourceSelectionCriteria**: (Für die Filterung erforderlich) Verwenden Sie `include or exclude` , um eine in resourceMatchers definierte Ressource ein- oder auszuschließen. Fügen Sie die folgenden resourceMatchers-Parameter hinzu, um die Ressourcen zu definieren, die ein- oder auszuschließen sind:
     - **resourceFilter.resourceMatchers**: Ein Array von resourceMatcher-Objekten. Wenn Sie mehrere Elemente in diesem Array definieren, werden diese mit einer ODER-Verknüpfung verglichen und die Felder innerhalb jedes Elements (group, kind, version) werden mit einer UND-Verknüpfung verglichen.
       - **resourceMatchers[].group**: (*Optional*) Gruppe der zu filternden Ressource.

- **resourceMatchers[].kind**: (*Optional*) Art der zu filternden Ressource.
- **resourceMatchers[].version**: (*Optional*) Version der zu filternden Ressource.
- **resourceMatchers[].names**: (*Optional*) Namen im Kubernetes metadata.name-Feld der Ressource, die gefiltert werden soll.
- **resourceMatchers[].namespaces**: (*Optional*) Namespaces im Kubernetes metadata.name-Feld der Ressource, die gefiltert werden soll.
- **resourceMatchers[].labelSelectors**: (*Optional*) Label-Selektorzeichenfolge im Kubernetes-Metadatenfeld name der Ressource, wie definiert in der "Kubernetes-Dokumentation". Zum Beispiel: `"trident.netapp.io/os=linux"`.

  Beispiel:

```
spec:
  resourceFilter:
    resourceSelectionCriteria: "include"
    resourceMatchers:
      - group: my-resource-group-1
        kind: my-resource-kind-1
        version: my-resource-version-1
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
      - group: my-resource-group-2
        kind: my-resource-kind-2
        version: my-resource-version-2
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
```

4. Nachdem Sie die `trident-protect-snapshot-restore-cr.yaml` Datei mit den korrekten Werten gefüllt haben, wenden Sie die CR an:

```
kubectl apply -f trident-protect-snapshot-restore-cr.yaml
```

Stellen Sie den Snapshot mithilfe der CLI wieder her.

Schritte

1. Stellen Sie den Snapshot in einem anderen Namespace wieder her, wobei Sie die Werte in Klammern durch Informationen aus Ihrer Umgebung ersetzen.

   - Das `snapshot` Argument verwendet einen Namespace und einen Snapshot-Namen im Format `<namespace>/<name>`.
   - Das `namespace-mapping` Argument verwendet durch Doppelpunkte getrennte Namensräume, um Quell-Namensräume den korrekten Ziel-Namensräumen im Format `source1:dest1,source2:dest2` zuzuordnen.

Beispiel:

```
tridentctl-protect create snapshotrestore <my_restore_name>
--snapshot <namespace/snapshot_to_restore> --namespace-mapping
<source_to_destination_namespace_mapping>
```

Trident Protect-Ausführungshooks verwalten

Ein Ausführungshook ist eine benutzerdefinierte Aktion, die Sie so konfigurieren können, dass sie in Verbindung mit einem Datenschutzvorgang einer verwalteten App ausgeführt wird. Wenn Sie beispielsweise eine Datenbank-App haben, können Sie einen Ausführungshook verwenden, um alle Datenbanktransaktionen vor einem Snapshot anzuhalten und die Transaktionen nach Abschluss des Snapshots fortzusetzen. Dies gewährleistet applikationskonsistente Snapshots.

Arten von Execution Hooks

Trident Protect unterstützt die folgenden Arten von Ausführungshooks, basierend darauf, wann sie ausgeführt werden können:

- Vor-Snapshot
- Nach dem Schnappschuss
- Pre-Backup
- Nach dem Backup
- Nach der Wiederherstellung
- Nach dem Failover

Ausführungsreihenfolge

Wenn ein Datenschutzvorgang ausgeführt wird, finden die Ausführungs-Hook-Ereignisse in der folgenden Reihenfolge statt:

1. Alle anwendbaren benutzerdefinierten Pre-Operation-Hooks werden in den entsprechenden Containern ausgeführt. Sie können beliebig viele benutzerdefinierte Pre-Operation-Hooks erstellen und ausführen, aber die Ausführungsreihenfolge dieser Hooks vor der Operation ist weder garantiert noch konfigurierbar.
2. Gegebenenfalls kommt es zu Dateisystem-Freezes. "Erfahren Sie mehr über die Konfiguration des Einfrierens von Dateisystemen mit Trident Protect".
3. Der Datenschutzvorgang wird durchgeführt.
4. Eingefrorene Dateisysteme werden, falls zutreffend, wieder freigegeben.
5. Alle anwendbaren benutzerdefinierten Nachbearbeitungs-Hooks werden in den entsprechenden Containern ausgeführt. Sie können beliebig viele benutzerdefinierte Nachbearbeitungs-Hooks erstellen und ausführen, aber die Ausführungsreihenfolge dieser Hooks nach der Operation ist weder garantiert noch konfigurierbar.

Wenn Sie mehrere Ausführungs-Hooks desselben Typs erstellen (z. B. pre-snapshot), ist die

Ausführungsreihenfolge dieser Hooks nicht garantiert. Die Ausführungsreihenfolge von Hooks unterschiedlicher Typen ist jedoch garantiert. Zum Beispiel ist dies die Ausführungsreihenfolge einer Konfiguration, die alle verschiedenen Hook-Typen enthält:

1. Pre-Snapshot-Hooks ausgeführt
2. Post-Snapshot-Hooks ausgeführt
3. Pre-Backup-Hooks ausgeführt
4. Post-Backup-Hooks ausgeführt

HINWEIS	Das vorhergehende Befehlsbeispiel gilt nur, wenn Sie eine Sicherung durchführen, die keinen vorhandenen Snapshot verwendet.
HINWEIS	Sie sollten Ihre Ausführungshook-Skripte immer testen, bevor Sie sie in einer Produktionsumgebung aktivieren. Sie können den Befehl 'kubectl exec' verwenden, um die Skripte bequem zu testen. Nachdem Sie die Ausführungshooks in einer Produktionsumgebung aktiviert haben, testen Sie die resultierenden Snapshots und Backups, um sicherzustellen, dass sie konsistent sind. Sie können dies tun, indem Sie die App in einen temporären Namespace klonen, den Snapshot oder das Backup wiederherstellen und dann die App testen.
HINWEIS	Wenn ein Pre-Snapshot-Ausführungs-Hook Kubernetes-Ressourcen hinzufügt, ändert oder entfernt, werden diese Änderungen in den Snapshot oder das Backup und in jede nachfolgende Wiederherstellungsoperation einbezogen.

Wichtige Hinweise zu benutzerdefinierten Ausführungshooks

Beachten Sie Folgendes bei der Planung von Execution Hooks für Ihre Apps.

- Ein Ausführungs-Hook muss ein Skript verwenden, um Aktionen auszuführen. Viele Ausführungs-Hooks können auf dasselbe Skript verweisen.
- Trident Protect erfordert, dass die Skripte, die von den Execution Hooks verwendet werden, im Format ausführbarer Shell-Skripte geschrieben sind.
- Die Skriptgröße ist auf 96KB begrenzt.
- Trident Protect verwendet Ausführungs-Hook-Einstellungen und alle übereinstimmenden Kriterien, um zu bestimmen, welche Hooks für einen Snapshot-, Backup- oder Wiederherstellungsvorgang anwendbar sind.

HINWEIS	Da Ausführungshooks die Funktionalität der Anwendung, auf der sie ausgeführt werden, oft einschränken oder vollständig deaktivieren, sollten Sie stets versuchen, die Zeit zu minimieren, die Ihre benutzerdefinierten Ausführungshooks zum Ausführen benötigen. Wenn Sie einen Sicherungs- oder Snapshot-Vorgang mit zugehörigen Ausführungshooks starten, diesen aber abbrechen, dürfen die Hooks dennoch ausgeführt werden, falls der Sicherungs- oder Snapshot-Vorgang bereits begonnen hat. Das bedeutet, dass die Logik, die in einem Ausführungshook nach der Sicherung verwendet wird, nicht davon ausgehen kann, dass die Sicherung abgeschlossen wurde.

Ausführungs-Hook-Filter

Wenn Sie einen Ausführungshook für eine Anwendung hinzufügen oder bearbeiten, können Sie Filter zum Ausführungshook hinzufügen, um zu steuern, auf welche Container der Hook angewendet wird. Filter sind nützlich für Anwendungen, die auf allen Containern dasselbe Container-Image verwenden, aber jedes Image

für einen anderen Zweck nutzen (wie zum Beispiel Elasticsearch). Filter ermöglichen es Ihnen, Szenarien zu erstellen, in denen Ausführungshooks auf einigen, aber nicht unbedingt allen identischen Containern ausgeführt werden. Wenn Sie mehrere Filter für einen einzelnen Ausführungshook erstellen, werden diese mit einem logischen UND-Operator kombiniert. Sie können bis zu 10 aktive Filter pro Ausführungshook haben.

Jeder Filter, den Sie einem Ausführungshook hinzufügen, verwendet einen regulären Ausdruck, um Container in Ihrem Cluster zu finden. Wenn ein Hook mit einem Container übereinstimmt, führt der Hook das zugehörige Skript auf diesem Container aus. Reguläre Ausdrücke für Filter verwenden die Regular Expression 2 (RE2) Syntax, die das Erstellen eines Filters, der Container von der Liste der Treffer ausschließt, nicht unterstützt. Weitere Informationen zur Syntax, die Trident Protect für reguläre Ausdrücke in Ausführungshook-Filtern unterstützt, finden Sie unter "Unterstützung der Regular Expression 2 (RE2) Syntax".

HINWEIS	Wenn Sie einem Ausführungshook, der nach einem Wiederherstellungs- oder Klonvorgang ausgeführt wird, einen Namespace-Filter hinzufügen und sich die Quelle und das Ziel des Wiederherstellungs- oder Klonvorgangs in unterschiedlichen Namespaces befinden, wird der Namespace-Filter nur auf den Ziel-Namespace angewendet.

Beispiele für Execution Hooks

Besuchen Sie das "NetApp Verda GitHub-Projekt", um echte Ausführungs-Hooks für beliebte Apps wie Apache Cassandra und Elasticsearch herunterzuladen. Sie können auch Beispiele sehen und Ideen für die Strukturierung Ihrer eigenen benutzerdefinierten Ausführungs-Hooks erhalten.

Erstellen Sie einen Ausführungs-Hook

Sie können einen benutzerdefinierten Ausführungs-Hook für eine App mit Trident Protect erstellen. Sie benötigen die Berechtigungen „Besitzer“, „Administrator“ oder „Mitglied“, um Ausführungs-Hooks zu erstellen.

Verwenden Sie einen CR

Schritte

1. Erstellen Sie die benutzerdefinierte Ressourcendatei (CR) und benennen Sie sie `trident-protect-hook.yaml`.
2. Konfigurieren Sie die folgenden Attribute so, dass sie Ihrer Trident Protect-Umgebung und Clusterkonfiguration entsprechen:

   - **metadata.name**: (*Erforderlich*) Der Name dieser benutzerdefinierten Ressource; wählen Sie einen eindeutigen und sinnvollen Namen für Ihre Umgebung.
   - **spec.applicationRef**: (*Erforderlich*) Der Kubernetes-Name der Anwendung, für die der Ausführungshook ausgeführt werden soll.
   - **spec.stage**: (*Erforderlich*) Eine Zeichenkette, die angibt, in welcher Phase der Aktion der Ausführungs-Hook ausgeführt werden soll. Mögliche Werte:
     - Vor
     - Beitrag
   - **spec.action**: (*Erforderlich*) Eine Zeichenkette, die angibt, welche Aktion der Ausführungs-Hook ausführt, vorausgesetzt, dass alle angegebenen Ausführungs-Hook-Filter übereinstimmen. Mögliche Werte:
     - Schnappschuss
     - Backup
     - Wiederherstellen
     - Failover
   - **spec.enabled**: (*Optional*) Gibt an, ob dieser Ausführungs-Hook aktiviert oder deaktiviert ist. Wenn nicht angegeben, ist der Standardwert true.
   - **spec.hookSource**: (*Erforderlich*) Eine Zeichenkette, die das Base64-kodierte Hook-Skript enthält.
   - **spec.timeout**: (*Optional*) Eine Zahl, die angibt, wie lange in Minuten der Ausführungs-Hook ausgeführt werden darf. Der Mindestwert beträgt 1 Minute und der Standardwert beträgt 25 Minuten, falls nicht angegeben.
   - **spec.arguments**: (*Optional*) Eine YAML-Liste von Argumenten, die Sie für den Ausführungshook angeben können.
   - **spec.matchingCriteria**: (*Optional*) Eine optionale Liste von Kriterien-Schlüssel-Wert-Paaren, wobei jedes Paar einen Ausführungs-Hook-Filter bildet. Sie können bis zu 10 Filter pro Ausführungs-Hook hinzufügen.
   - **spec.matchingCriteria.type**: (*Optional*) Eine Zeichenkette, die den Filtertyp des Ausführungshooks identifiziert. Mögliche Werte:
     - ContainerImage
     - ContainerName
     - PodName
     - PodLabel
     - NamespaceName
   - **spec.matchingCriteria.value**: (*Optional*) Eine Zeichenkette oder ein regulärer Ausdruck, der den Wert des Ausführungs-Hook-Filters identifiziert.

Beispiel YAML:

```
apiVersion: protect.trident.netapp.io/v1
kind: ExecHook
metadata:
  name: example-hook-cr
  namespace: my-app-namespace
  annotations:
    astra.netapp.io/astra-control-hook-source-id:
/account/test/hookSource/id
spec:
  applicationRef: my-app-name
  stage: Pre
  action: Snapshot
  enabled: true
  hookSource: IyEvYmluL2Jhc2gKZWNobyAiZXhhbXBsZSBzY3JpcHQiCg==
  timeout: 10
  arguments:
    - FirstExampleArg
    - SecondExampleArg
  matchingCriteria:
    - type: containerName
      value: mysql
    - type: containerImage
      value: bitnami/mysql
    - type: podName
      value: mysql
    - type: namespaceName
      value: mysql-a
    - type: podLabel
      value: app.kubernetes.io/component=primary
    - type: podLabel
      value: helm.sh/chart=mysql-10.1.0
    - type: podLabel
      value: deployment-type=production
```

3. Nachdem Sie die CR-Datei mit den korrekten Werten gefüllt haben, wenden Sie die CR an:

```
kubectl apply -f trident-protect-hook.yaml
```

Verwenden Sie die Befehlszeile

Schritte

1. Erstellen Sie den Ausführungs-Hook, und ersetzen Sie die Werte in Klammern durch Informationen aus Ihrer Umgebung. Beispiel:

```
tridentctl-protect create exechook <my_exec_hook_name> --action
<action_type> --app <app_to_use_hook> --stage <pre_or_post_stage>
--source-file <script-file> -n <application_namespace>
```

Einen Ausführungshook manuell ausführen

Sie können einen Ausführungs-Hook manuell zu Testzwecken ausführen oder wenn Sie den Hook nach einem Fehler manuell erneut ausführen müssen. Sie benötigen die Berechtigungen „Besitzer“, „Administrator“ oder „Mitglied“, um Ausführungs-Hooks manuell auszuführen.

Das manuelle Ausführen eines Execution Hooks besteht aus zwei grundlegenden Schritten:

1. Erstellen Sie eine Ressourcensicherung, die Ressourcen sammelt und eine Sicherung davon erstellt, wobei festgelegt wird, wo der Hook ausgeführt wird
2. Führe den Ausführungshook gegen das Backup aus

Schritt 1: Erstellen Sie eine Ressourcensicherung

Verwenden Sie einen CR

Schritte

1. Erstellen Sie die benutzerdefinierte Ressourcendatei (CR) und benennen Sie sie `trident-protect-resource-backup.yaml`.
2. Konfigurieren Sie die folgenden Attribute so, dass sie Ihrer Trident Protect-Umgebung und Clusterkonfiguration entsprechen:
   - **metadata.name**: (*Erforderlich*) Der Name dieser benutzerdefinierten Ressource; wählen Sie einen eindeutigen und sinnvollen Namen für Ihre Umgebung.
   - **spec.applicationRef**: (*Erforderlich*) Der Kubernetes-Name der Anwendung, für die das Ressourcen-Backup erstellt werden soll.
   - **spec.appVaultRef**: (*Erforderlich*) Der Name des AppVault, in dem die Sicherungsinhalte gespeichert sind.
   - **spec.appArchivePath**: Der Pfad innerhalb von AppVault, in dem die Sicherungsinhalte gespeichert sind. Sie können den folgenden Befehl verwenden, um diesen Pfad zu finden:

     ```
     kubectl get backups <BACKUP_NAME> -n my-app-namespace -o jsonpath='{.status.appArchivePath}'
     ```

     Beispiel YAML:

     ```
     ---
     apiVersion: protect.trident.netapp.io/v1
     kind: ResourceBackup
     metadata:
       name: example-resource-backup
     spec:
       applicationRef: my-app-name
       appVaultRef: my-appvault-name
       appArchivePath: example-resource-backup
     ```

3. Nachdem Sie die CR-Datei mit den korrekten Werten gefüllt haben, wenden Sie die CR an:

   ```
   kubectl apply -f trident-protect-resource-backup.yaml
   ```

Verwenden Sie die Befehlszeile

Schritte

1. Erstellen Sie die Sicherungskopie, indem Sie die Werte in Klammern durch Informationen aus Ihrer Umgebung ersetzen. Beispiel:

```
tridentctl protect create resourcebackup <my_backup_name> --app
<my_app_name> --appvault <my_appvault_name> -n
<my_app_namespace> --app-archive-path <app_archive_path>
```

2. Zeigen Sie den Status der Datensicherung an. Sie können diesen Beispielbefehl wiederholt verwenden, bis der Vorgang abgeschlossen ist:

```
tridentctl protect get resourcebackup -n <my_app_namespace>
<my_backup_name>
```

3. Überprüfen Sie, ob das Backup erfolgreich war:

```
kubectl describe resourcebackup <my_backup_name>
```

Schritt 2: Den Ausführungshook ausführen

Verwenden Sie einen CR

Schritte

1. Erstellen Sie die benutzerdefinierte Ressourcendatei (CR) und benennen Sie sie `trident-protect-hook-run.yaml`.
2. Konfigurieren Sie die folgenden Attribute so, dass sie Ihrer Trident Protect-Umgebung und Clusterkonfiguration entsprechen:
   - **metadata.name**: (*Erforderlich*) Der Name dieser benutzerdefinierten Ressource; wählen Sie einen eindeutigen und sinnvollen Namen für Ihre Umgebung.
   - **spec.applicationRef**: (*Erforderlich*) Stellen Sie sicher, dass dieser Wert mit dem Anwendungsnamen aus dem ResourceBackup CR übereinstimmt, den Sie in Schritt 1 erstellt haben.
   - **spec.appVaultRef**: (*Erforderlich*) Stellen Sie sicher, dass dieser Wert mit dem appVaultRef aus dem ResourceBackup CR übereinstimmt, den Sie in Schritt 1 erstellt haben.
   - **spec.appArchivePath**: Stellen Sie sicher, dass dieser Wert mit dem appArchivePath aus dem ResourceBackup CR übereinstimmt, den Sie in Schritt 1 erstellt haben.

     ```
     kubectl get backups <BACKUP_NAME> -n my-app-namespace -o
     jsonpath='{.status.appArchivePath}'
     ```

   - **spec.action**: (*Erforderlich*) Eine Zeichenkette, die angibt, welche Aktion der Ausführungs-Hook ausführt, vorausgesetzt, dass alle angegebenen Ausführungs-Hook-Filter übereinstimmen. Mögliche Werte:
     - Schnappschuss
     - Backup
     - Wiederherstellen
     - Failover
   - **spec.stage**: (*Erforderlich*) Eine Zeichenkette, die angibt, in welcher Phase der Aktion der Ausführungs-Hook ausgeführt werden soll. Dieser Hook-Lauf führt keine Hooks in anderen Phasen aus. Mögliche Werte:
     - Vor
     - Beitrag

     Beispiel YAML:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: ExecHooksRun
metadata:
  name: example-hook-run
spec:
  applicationRef: my-app-name
  appVaultRef: my-appvault-name
  appArchivePath: example-resource-backup
  stage: Post
  action: Failover
```

3. Nachdem Sie die CR-Datei mit den korrekten Werten gefüllt haben, wenden Sie die CR an:

```
kubectl apply -f trident-protect-hook-run.yaml
```

Verwenden Sie die Befehlszeile

Schritte

1. Erstellen Sie die manuelle Ausführungs-Hook-Anforderung:

```
tridentctl protect create exechooksrun <my_exec_hook_run_name>
-n <my_app_namespace> --action snapshot --stage <pre_or_post>
--app <my_app_name> --appvault <my_appvault_name> --path
<my_backup_name>
```

2. Überprüfen Sie den Status des ausgeführten Hooks. Sie können diesen Befehl so lange wiederholen, bis der Vorgang abgeschlossen ist:

```
tridentctl protect get exechooksrun -n <my_app_namespace>
<my_exec_hook_run_name>
```

3. Beschreiben Sie das exechooksrun-Objekt, um die endgültigen Details und den Status anzuzeigen:

```
kubectl -n <my_app_namespace> describe exechooksrun
<my_exec_hook_run_name>
```

Trident Protect deinstallieren

Möglicherweise müssen Sie Trident Protect-Komponenten entfernen, wenn Sie von einer Testversion auf eine Vollversion des Produkts aktualisieren.

Um Trident Protect zu entfernen, führen Sie die folgenden Schritte aus.

Schritte

1. Entfernen Sie die Trident Protect CR-Dateien:

   HINWEIS Dieser Schritt ist für Version 25.06 und höher nicht erforderlich.

   ```
   helm uninstall -n trident-protect trident-protect-crds
   ```

2. Trident Protect entfernen:

   ```
   helm uninstall -n trident-protect trident-protect
   ```

3. Entfernen Sie den Trident Protect-Namespace:

   ```
   kubectl delete ns trident-protect
   ```

Trident- und Trident Protect-Blogs

Hier finden Sie einige großartige NetApp Trident und Trident Protect Blogs:

Trident Blogs

- 16. Oktober 2025: "Erweiterte Speicherlösungen für Kubernetes"
- 19. August 2025: "Verbesserung der Datenkonsistenz: Volume Group Snapshots in OpenShift Virtualisierung mit Trident"
- 9. Mai 2025: "Automatische Trident Backend-Konfiguration für FSx for ONTAP mit dem Amazon EKS Add-on"
- 15. April 2025: "NetApp Trident mit Google Cloud NetApp Volumes für SMB-Protokoll"
- 14. April 2025: "Nutzung des Fiber Channel Protocol mit Trident 25.02 für persistenten Speicher auf Kubernetes"
- 14. April 2025: "Die Leistungsfähigkeit von NetApp ASA r2-Systemen für Kubernetes-Blockspeicher erschließen"
- 31. März 2025: "Vereinfachung der Trident-Installation auf Red Hat OpenShift mit dem neuen zertifizierten Operator"
- 27. März 2025: "Bereitstellung von Trident für SMB mit Google Cloud NetApp Volumes"
- 5. März 2025: "Nahtlose iSCSI-Speicherintegration freischalten: Ein Leitfaden zu FSxN auf ROSA Clusters für AWS"
- 27. Februar 2025: "Bereitstellung von Cloud-Identität mit Trident, GKE und Google Cloud NetApp Volumes"
- 12. Dezember 2024: "Einführung der Fibre Channel-Unterstützung in Trident"
- 11. November 2024: "NetApp Trident mit Google Cloud NetApp Volumes"
- 29. Oktober 2024: "Amazon FSx for NetApp ONTAP mit Red Hat OpenShift Service on AWS (ROSA) unter Verwendung von Trident"
- 29. Oktober 2024: "Live-Migration von VMs mit OpenShift Virtualisierung auf ROSA und Amazon FSx für NetApp ONTAP"
- 8. Juli 2024: "Verwendung von NVMe/TCP zur Nutzung von ONTAP Storage für Ihre modernen containerisierten Apps auf Amazon EKS"
- 1. Juli 2024: "Nahtloser Kubernetes-Speicher mit Google Cloud NetApp Volumes Flex und Astra Trident"
- 11. Juni 2024: "ONTAP als Backend-Speicher für die integrierte Bildregistrierung in OpenShift"

Trident Protect Blogs

- 16. Mai 2025: "Automatisierung des Registry-Failover für die Notfallwiederherstellung mit Trident Protect post-restore hooks"
- 16. Mai 2025: "OpenShift Virtualisierung Notfallwiederherstellung mit NetApp Trident Protect"
- 13. Mai 2025: "Migration von Storage-Klassen mit Trident Protect Backup Restore"
- 9. Mai 2025: "Kubernetes-Anwendungen mit Trident Protect Post-Restore-Hooks neu skalieren"
- 03. April 2025: "Trident Protect Power Up: Kubernetes-Replikation für Schutz Notfallwiederherstellung"
- 13. März 2025: "Absturzsichere Sicherungs- und Wiederherstellungsvorgänge für OpenShift

Virtualisierungs-VMs"

- 11. März 2025: "Erweiterung von GitOps-Mustern auf den Anwendungsdatenschutz mit NetApp Trident"
- 03. März 2025: "Trident 25.02: Verbesserung des Red Hat OpenShift Erlebnisses mit spannenden neuen Funktionen"
- 15. Januar 2025: "Einführung der rollenbasierten Zugriffssteuerung von Trident Protect"
- 11. November 2024: "Kubernetes-gesteuertes Datenmanagement: Die neue Ära mit Trident Protect"

Wissen und Unterstützung

Häufig gestellte Fragen

Hier finden Sie Antworten auf häufig gestellte Fragen zur Installation, Konfiguration, Aktualisierung und Fehlerbehebung von Trident.

Allgemeine Fragen

Wie häufig wird Trident veröffentlicht?

Ab der Version 24.02 wird Trident alle vier Monate veröffentlicht: Februar, Juni und Oktober.

Unterstützt Trident alle Funktionen, die in einer bestimmten Version von Kubernetes veröffentlicht werden?

Trident unterstützt in Kubernetes üblicherweise keine Alpha-Funktionen. Trident könnte Beta-Funktionen innerhalb der zwei Trident-Versionen unterstützen, die auf die Kubernetes-Beta-Version folgen.

Hat Trident Abhängigkeiten von anderen NetApp Produkten für seine Funktionsweise?

Trident hat keine Abhängigkeiten von anderen NetApp Softwareprodukten und funktioniert als eigenständige Anwendung. Sie sollten jedoch ein NetApp Backend-Speichergerät haben.

Wie kann ich vollständige Trident-Konfigurationsdetails erhalten?

Verwenden Sie den `tridentctl get` Befehl, um weitere Informationen zu Ihrer Trident-Konfiguration zu erhalten.

Kann ich Kennzahlen darüber erhalten, wie Speicher von Trident bereitgestellt wird?

Ja. Prometheus-Endpunkte können verwendet werden, um Informationen über den Betrieb von Trident zu erfassen, wie die Anzahl der verwalteten Backends, die Anzahl der bereitgestellten Volumes, den Datenverbrauch in Bytes und so weiter. Sie können auch "Cloud Insights" für Überwachung und Analyse verwenden.

Verändert sich die Benutzererfahrung bei der Verwendung von Trident als CSI-Provisioner?

Nein. Hinsichtlich Benutzerfreundlichkeit und Funktionalität gibt es keine Änderungen. Der verwendete Provisionierungsname ist `csi.trident.netapp.io`. Diese Methode der Installation von Trident wird empfohlen, wenn Sie alle neuen Funktionen aktueller und zukünftiger Versionen nutzen möchten.

Installieren und verwenden Sie Trident auf einem Kubernetes-Cluster

Unterstützt Trident eine Offline-Installation aus einer privaten Registry?

Ja, Trident kann offline installiert werden. Siehe "Erfahren Sie mehr über die Trident-Installation".

Kann ich Trident remote installieren?

Ja. Trident 18.10 und höher unterstützen die Remote-Installationsfunktion von jedem Rechner, der `kubectl` Zugriff auf den Cluster hat. Nachdem `kubectl` der Zugriff verifiziert wurde (zum Beispiel, indem Sie einen

`kubectl get nodes` Befehl vom Remote-Rechner initiieren, um dies zu überprüfen), folgen Sie den Installationsanweisungen.

Kann ich Hochverfügbarkeit mit Trident konfigurieren?

Trident wird als Kubernetes Deployment (ReplicaSet) mit einer Instanz installiert und verfügt daher über integrierte Hochverfügbarkeit. Sie sollten die Anzahl der Replikate in der Implementierung nicht erhöhen. Wenn der Knoten, auf dem Trident installiert ist, verloren geht oder der Pod anderweitig nicht zugänglich ist, stellt Kubernetes den Pod automatisch auf einem gesunden Knoten in Ihrem Cluster erneut bereit. Trident ist nur Steuerungsebene, daher sind aktuell eingebundene Pods nicht betroffen, wenn Trident neu bereitgestellt wird.

Benötigt Trident Zugriff auf den Namespace kube-system?

Trident liest vom Kubernetes API Server, um festzustellen, wann Anwendungen neue PVCs anfordern, daher benötigt es Zugriff auf kube-system.

Welche Rollen und Privilegien werden von Trident verwendet?

Der Trident-Installer erstellt eine Kubernetes-ClusterRole, die spezifischen Zugriff auf die PersistentVolume-, PersistentVolumeClaim-, StorageClass- und Secret-Ressourcen des Kubernetes-Clusters hat. Siehe "tridentctl-Installation anpassen".

Kann ich die exakten Manifestdateien, die Trident für die Installation verwendet, lokal generieren?

Sie können die von Trident für die Installation verwendeten Manifestdateien bei Bedarf lokal generieren und bearbeiten. Siehe "tridentctl-Installation anpassen".

Kann ich dieselbe ONTAP Backend-SVM für zwei separate Trident Instanzen für zwei separate Kubernetes-Cluster verwenden?

Obwohl es nicht empfohlen wird, können Sie dieselbe Backend-SVM für zwei Trident-Instanzen verwenden. Geben Sie während der Installation für jede Instanz einen eindeutigen Volume-Namen an und/oder definieren Sie einen eindeutigen `StoragePrefix` Parameter in der `setup/backend.json` Datei. Dies soll sicherstellen, dass nicht dasselbe FlexVol Volume für beide Instanzen verwendet wird.

Ist es möglich, Trident unter ContainerLinux (formerly CoreOS) zu installieren?

Trident ist einfach ein Kubernetes pod und kann überall dort installiert werden, wo Kubernetes läuft.

Kann ich Trident mit NetApp Cloud Volumes ONTAP verwenden?

Ja, Trident wird auf AWS, Google Cloud und Azure unterstützt.

Fehlerbehebung und Support

Unterstützt NetApp Trident?

Obwohl Trident Open Source ist und kostenlos zur Verfügung gestellt wird, wird es von NetApp vollständig unterstützt, sofern Ihr NetApp Backend unterstützt wird.

Wie kann ich einen Support-Fall eröffnen?

Um einen Supportfall zu eröffnen, führen Sie einen der folgenden Schritte aus:

1. Wenden Sie sich an Ihren Support Account Manager und lassen Sie sich bei der Erstellung eines Tickets helfen.
2. Eröffnen Sie einen Supportfall, indem Sie "NetApp Support" kontaktieren.

Wie erstelle ich ein Support-Log-Bundle?

Sie können ein Support-Bundle erstellen, indem Sie `tridentctl logs -a` ausführen. Zusätzlich zu den im Bundle erfassten Protokollen erfassen Sie das Kubelet-Protokoll, um die Mount-Probleme auf der Kubernetes-Seite zu diagnostizieren. Die Anweisungen zum Abrufen des Kubelet-Protokolls variieren je nachdem, wie Kubernetes installiert ist.

Was muss ich tun, wenn ich eine Anfrage für ein neues Feature stellen möchte?

Erstellen Sie ein Ticket auf "Trident Github" und erwähnen Sie **RFE** im Betreff und in der Beschreibung des Tickets.

Wo kann ich einen Defekt melden?

Erstellen Sie ein Ticket auf "Trident Github". Stellen Sie sicher, dass Sie alle notwendigen Informationen und Protokolle zu dem Problem angeben.

Was passiert, wenn ich eine kurze Frage zu Trident habe, die ich klären möchte? Gibt es eine Community oder ein Forum?

Bei Fragen, Problemen oder Anliegen kontaktieren Sie uns bitte über unseren Trident "Discord-Kanal" oder GitHub.

Das Passwort meines Speichersystems wurde geändert und Trident funktioniert nicht mehr, wie kann ich es wiederherstellen?

Aktualisieren Sie das Passwort des Backends mit `tridentctl update backend myBackend -f </path/to_new_backend.json> -n trident`. Ersetzen Sie `myBackend` im Beispiel durch Ihren Backend-Namen und `/path/to_new_backend.json` durch den Pfad zur richtigen `backend.json` Datei.

Trident kann meinen Kubernetes-Knoten nicht finden. Wie behebe ich das?

Es gibt zwei wahrscheinliche Szenarien, warum Trident einen Kubernetes-Knoten nicht finden kann. Dies kann an einem Netzwerkproblem innerhalb von Kubernetes oder an einem DNS-Problem liegen. Das Trident-Knoten-Daemonset, das auf jedem Kubernetes-Knoten läuft, muss mit dem Trident-Controller kommunizieren können, um den Knoten bei Trident zu registrieren. Wenn nach der Installation von Trident Netzwerkänderungen vorgenommen wurden, tritt dieses Problem nur bei neuen Kubernetes-Knoten auf, die dem Cluster hinzugefügt werden.

Wenn das Trident-Pod zerstört wird, gehen die Daten verloren?

Daten gehen nicht verloren, wenn der Trident-Pod zerstört wird. Trident-Metadaten werden in CRD-Objekten gespeichert. Alle von Trident bereitgestellten PVs funktionieren weiterhin normal.

Trident aktualisieren

Kann ich direkt von einer älteren Version auf eine neuere Version aktualisieren (einige Versionen überspringen)?

NetApp unterstützt das Upgrade von Trident von einer Hauptversion auf die nächstfolgende Hauptversion. Sie können von Version 18.xx auf 19.xx, von 19.xx auf 20.xx und so weiter aktualisieren. Sie sollten das Upgrade in einer Testumgebung testen, bevor Sie die Implementierung in der Produktion durchführen.

Ist es möglich, Trident auf eine frühere Version zurückzusetzen?

Falls nach einem Upgrade Fehler, Abhängigkeitsprobleme oder ein fehlgeschlagenes bzw. unvollständiges Upgrade auftreten, sollten Sie "Trident deinstallieren" und die vorherige Version gemäß der entsprechenden Anleitung für diese Version neu installieren. Dies ist die einzige empfohlene Methode, um auf eine frühere Version zurückzukehren.

Backends und Volumes verwalten

Muss ich sowohl Management- als auch DataLIFs in einer ONTAP Backend-Definitionsdatei definieren?

Das Management-LIF ist obligatorisch. Das DataLIF variiert:

- ONTAP SAN: Nicht für iSCSI angeben. Trident verwendet "ONTAP Selective LUN Map" zur Erkennung der iSCSI-LIFs, die für die Einrichtung einer Multipath-Sitzung benötigt werden. Eine Warnung wird generiert, wenn `dataLIF` explizit definiert ist. Siehe "ONTAP SAN-Konfigurationsoptionen und Beispiele" für Details.
- ONTAP NAS: NetApp empfiehlt, `dataLIF` anzugeben. Falls nicht angegeben, ruft Trident dataLIFs vom SVM ab. Ein vollqualifizierter Domainname (FQDN) kann für die NFS-Mount-Operationen angegeben werden, wodurch die Erstellung eines Round-Robin-DNS zur Lastverteilung über mehrere dataLIFs ermöglicht wird. Weitere Informationen finden Sie unter "ONTAP NAS-Konfigurationsoptionen und Beispiele".

Kann Trident CHAP für ONTAP Backends konfigurieren?

Ja. Trident unterstützt bidirektionales CHAP für ONTAP Backends. Dies erfordert das Setzen von `useCHAP=true` in Ihrer Backend-Konfiguration.

Wie verwalte ich Export-Richtlinien mit Trident?

Trident kann ab Version 20.04 Exportrichtlinien dynamisch erstellen und verwalten. Dies ermöglicht es dem Speicheradministrator, in seiner Backend-Konfiguration einen oder mehrere CIDR-Blöcke anzugeben und Trident fügt die Knoten-IPs, die in diese Bereiche fallen, zu einer von ihm erstellten Exportrichtlinie hinzu. Auf diese Weise verwaltet Trident automatisch das Hinzufügen und Löschen von Regeln für Knoten mit IPs innerhalb der angegebenen CIDRs.

Können IPv6-Adressen für die Management- und DataLIFs verwendet werden?

Trident unterstützt die Definition von IPv6-Adressen für:

- `managementLIF` and `dataLIF` für ONTAP NAS-Backends.
- `managementLIF` für ONTAP SAN backends. Sie können `dataLIF` auf einem ONTAP SAN-Backend nicht angeben.

Trident muss mit dem Flag `--use-ipv6` (für `tridentctl` Installation), `IPv6` (für Trident operator) oder `tridentTPv6` (für Helm installation) installiert werden, damit es über IPv6 funktioniert.

Ist es möglich, das Management-LIF im Backend zu aktualisieren?

Ja, es ist möglich, das Backend-Management-LIF mit dem `tridentctl update backend` Befehl zu aktualisieren.

Ist es möglich, die DataLIF im Backend zu aktualisieren?

Sie können die DataLIF auf `ontap-nas` und `ontap-nas-economy`nur aktualisieren.

Kann ich mehrere Backends in Trident für Kubernetes erstellen?

Trident kann viele Backends gleichzeitig unterstützen, entweder mit dem gleichen Treiber oder mit verschiedenen Treibern.

Wie speichert Trident die Backend-Zugangsdaten?

Trident speichert die Backend-Zugangsdaten als Kubernetes Secrets.

Wie wählt Trident ein bestimmtes Backend aus?

Falls die Backend-Attribute nicht verwendet werden können, um automatisch die richtigen Pools für eine Klasse auszuwählen, werden die `storagePools` und `additionalStoragePools` Parameter verwendet, um einen bestimmten Satz von Pools auszuwählen.

Wie kann ich sicherstellen, dass Trident nicht von einem bestimmten Backend bereitstellt?

Der `excludeStoragePools`Parameter wird verwendet, um die Menge der Pools zu filtern, die Trident für die Bereitstellung verwendet, und entfernt alle Pools, die übereinstimmen.

Wenn mehrere Backends desselben Typs vorhanden sind, wie wählt Trident das zu verwendende Backend aus?

Wenn mehrere Backends desselben Typs konfiguriert sind, wählt Trident das passende Backend basierend auf den Parametern in `StorageClass` und `PersistentVolumeClaim` aus. Wenn zum Beispiel mehrere ontap-nas Driver Backends vorhanden sind, versucht Trident, die Parameter in `StorageClass` und `PersistentVolumeClaim` zusammen abzugleichen und ein Backend zu finden, das die Anforderungen, die in `StorageClass` und `PersistentVolumeClaim` aufgeführt sind, erfüllen kann. Wenn mehrere Backends die Anfrage erfüllen, wählt Trident zufällig eines davon aus.

Unterstützt Trident bidirektionales CHAP mit Element/SolidFire?

Ja.

Wie stellt Trident Qtrees auf einem ONTAP-Volume bereit? Wie viele Qtrees können auf einem einzelnen Volume bereitgestellt werden?

Der `ontap-nas-economy` Treiber erstellt bis zu 200 Qtrees im selben FlexVol Volume (konfigurierbar zwischen 50 und 300), 100.000 Qtrees pro Clusterknoten und 2,4M pro Cluster. Wenn Sie einen neuen `PersistentVolumeClaim` Qtree eingeben, der vom Economy-Treiber bedient wird, prüft der Treiber, ob bereits ein FlexVol Volume existiert, das den neuen Qtree bedienen kann. Wenn das FlexVol Volume, das den Qtree bedienen kann, nicht existiert, wird ein neues FlexVol Volume erstellt.

Wie kann ich Unix-Berechtigungen für auf ONTAP NAS bereitgestellte Volumes festlegen?

Sie können Unix-Berechtigungen für das von Trident bereitgestellte Volume festlegen, indem Sie einen Parameter in der Backend-Definitionsdatei setzen.

Wie kann ich beim Bereitstellen eines Volumes einen expliziten Satz von ONTAP NFS-Mount-Optionen konfigurieren?

Standardmäßig setzt Trident bei Kubernetes keine Mount-Optionen. Um die Mount-Optionen in der Kubernetes-Speicherklasse festzulegen, folgen Sie dem angegebenen Beispiel "hier,".

Wie kann ich die bereitgestellten Volumes auf eine bestimmte Exportrichtlinie einstellen?

Um den entsprechenden Hosts den Zugriff auf ein Volume zu ermöglichen, verwenden Sie den `exportPolicy` Parameter, der in der Backend-Definitionsdatei konfiguriert ist.

Wie stelle ich die Volumenverschlüsselung über Trident mit ONTAP ein?

Sie können die Verschlüsselung auf dem von Trident bereitgestellten Volume mithilfe des Verschlüsselungsparameters in der Backend-Definitionsdatei aktivieren. Weitere Informationen finden Sie unter: "Wie Trident mit NVE und NAE zusammenarbeitet"

Was ist der beste Weg, QoS für ONTAP über Trident zu implementieren?

Verwenden Sie `StorageClasses` zur Implementierung von QoS für ONTAP.

Wie gebe ich Thin oder Thick Provisioning über Trident an?

Die ONTAP-Treiber unterstützen entweder Thin oder Thick Provisioning. Die ONTAP-Treiber verwenden standardmäßig Thin Provisioning. Wenn Thick Provisioning gewünscht ist, sollten Sie entweder die Backend-Definitionsdatei oder die `StorageClass` konfigurieren. Wenn beide konfiguriert sind, hat `StorageClass` Vorrang. Konfigurieren Sie Folgendes für ONTAP:

1. Auf `StorageClass` das `provisioningType`-Attribut als dick setzen.
2. Aktivieren Sie in der Backend-Definitionsdatei Thick Volumes, indem Sie `backend spaceReserve parameter` als Volume festlegen.

Wie kann ich sicherstellen, dass die verwendeten Volumes nicht gelöscht werden, selbst wenn ich versehentlich das PVC lösche?

Der PVC-Schutz ist ab Kubernetes Version 1.10 automatisch aktiviert.

Kann ich NFS-PVCs, die von Trident erstellt wurden, vergrößern?

Ja. Sie können ein von Trident erstelltes PVC erweitern. Beachten Sie, dass die automatische Volumenvergrößerung eine ONTAP Funktion ist, die für Trident nicht anwendbar ist.

Kann ich ein Volume importieren, während es sich im SnapMirror Data Protection (DP) oder im Offline-Modus befindet?

Der Volume-Import schlägt fehl, wenn sich das externe Volume im DP-Modus befindet oder offline ist. Sie erhalten die folgende Fehlermeldung:

```
Error: could not import volume: volume import failed to get size of
volume: volume <name> was not found (400 Bad Request) command terminated
with exit code 1.
Make sure to remove the DP mode or put the volume online before importing
the volume.
```

Wie wird ein Ressourcenkontingent auf einen NetApp Cluster übertragen?

Kubernetes-Speicherressourcenkontingente sollten funktionieren, solange NetApp Storage über Kapazität verfügt. Wenn der NetApp Storage die Kubernetes-Kontingenteinstellungen aufgrund mangelnder Kapazität nicht einhalten kann, versucht Trident die Bereitstellung, schlägt jedoch fehl.

Kann ich mit Trident Volume Snapshots erstellen?

Ja. Das Erstellen von On-Demand-Volume-Snapshots und Persistent Volumes aus Snapshots wird von Trident unterstützt. Um PVs aus Snapshots zu erstellen, stelle sicher, dass das `VolumeSnapshotDataSource` Feature-Gate aktiviert wurde.

Welche Treiber unterstützen Trident Volume-Snapshots?

Ab heute ist die Unterstützung für On-Demand-Snapshots für unsere `ontap-nas`, `ontap-nas-flexgroup`, `ontap-san`, `ontap-san-economy`, `solidfire-san` und `azure-netapp-files` Backend-Treiber verfügbar.

Wie erstelle ich eine Snapshot-Sicherung eines von Trident mit ONTAP bereitgestellten Volumes?

Dies ist verfügbar auf `ontap-nas`, `ontap-san` und `ontap-nas-flexgroup`-Treibern. Sie können auch ein `snapshotPolicy` für den `ontap-san-economy`-Treiber auf der FlexVol-Ebene angeben.

Dies ist auch auf den `ontap-nas-economy` Treibern verfügbar, jedoch auf der FlexVol Volume-Ebene der Granularität und nicht auf der Qtree-Ebene der Granularität. Um die Möglichkeit zu aktivieren, Snapshots von durch Trident bereitgestellten Volumes zu erstellen, setzen Sie die Backend-Parameteroption `snapshotPolicy` auf die gewünschte Snapshot-Richtlinie, wie sie im ONTAP-Backend definiert ist. Alle vom Storage-Controller erstellten Snapshots sind Trident nicht bekannt.

Kann ich einen Snapshot-Reservierungsprozentsatz für ein über Trident bereitgestelltes Volume festlegen?

Ja, Sie können einen bestimmten Prozentsatz des Speicherplatzes für die Speicherung der Snapshot-Kopien über Trident reservieren, indem Sie das `snapshotReserve` Attribut in der Backend-Definitionsdatei festlegen. Wenn Sie `snapshotPolicy` und `snapshotReserve` in der Backend-Definitionsdatei konfiguriert haben, wird der Snapshot-Reservierungsprozentsatz entsprechend dem im Backend-File angegebenen `snapshotReserve` Prozentsatz festgelegt. Wenn die `snapshotReserve` Prozentzahl nicht angegeben ist, nimmt ONTAP standardmäßig den Snapshot-Reservierungsprozentsatz als 5. Wenn die `snapshotPolicy` Option auf none gesetzt ist, beträgt der Snapshot-Reservierungsprozentsatz 0.

Kann ich direkt auf das Volume-Snapshot-Verzeichnis zugreifen und Dateien kopieren?

Ja, Sie können auf das Snapshot-Verzeichnis auf dem von Trident bereitgestellten Volume zugreifen, indem Sie den `snapshotDir` Parameter in der Backend-Definitionsdatei festlegen.

Kann ich SnapMirror für Volumes über Trident einrichten?

Derzeit muss SnapMirror extern über die ONTAP CLI oder OnCommand System Manager eingerichtet werden.

Wie kann ich persistente Volumes auf einen bestimmten ONTAP-Snapshot wiederherstellen?

Um ein Volume auf einen ONTAP-Snapshot wiederherzustellen, führen Sie die folgenden Schritte aus:

1. Versetzen Sie den Anwendungspod, der das Persistent Volume verwendet, in den Ruhezustand.
2. Stellen Sie den erforderlichen Snapshot über die ONTAP CLI oder OnCommand System Manager wieder her.
3. Starten Sie den Anwendungspod neu.

Kann Trident Volumes auf SVMs bereitstellen, die mit einem Load-Sharing Mirror konfiguriert sind?

Für Root-Volumes von SVMs, die Daten über NFS bereitstellen, können Load-Sharing-Mirrors erstellt werden. ONTAP aktualisiert Load-Sharing-Mirrors automatisch für Volumes, die von Trident erstellt wurden. Dies kann zu Verzögerungen beim Mounten von Volumes führen. Wenn mehrere Volumes mit Trident erstellt werden, hängt die Bereitstellung eines Volumes davon ab, dass ONTAP den Load-Sharing-Mirror aktualisiert.

Wie kann ich die Speicherklassennutzung für jeden Kunden/Mandanten separat aufschlüsseln?

Kubernetes erlaubt keine Speicherklassen in Namespaces. Sie können jedoch Kubernetes verwenden, um die Nutzung einer bestimmten Speicherklasse pro Namespace mithilfe von Speicherressourcenquoten (Storage Resource Quotas), die pro Namespace gelten, einzuschränken. Um einem bestimmten Namespace den Zugriff auf einen bestimmten Speicher zu verweigern, setzen Sie die Ressourcenquote für diese Speicherklasse auf 0.

Fehlerbehebung

Nutzen Sie die hier bereitgestellten Hinweise zur Fehlerbehebung bei Problemen, die während der Installation und Verwendung von Trident auftreten können.

HINWEIS	Für Hilfe mit Trident erstellen Sie ein Support-Bundle mit `tridentctl logs -a -n trident` und senden Sie es an NetApp Support.

Allgemeine Fehlerbehebung

- Falls der Trident Pod nicht ordnungsgemäß startet (beispielsweise, wenn der Trident Pod in der `ContainerCreating` Phase mit weniger als zwei bereiten Containern festhängt), kann das Ausführen von `kubectl -n trident describe deployment trident` und `kubectl -n trident describe pod trident-**` zusätzliche Erkenntnisse liefern. Das Abrufen von kubelet-Logs (zum Beispiel über `journalctl -xeu kubelet`) kann ebenfalls hilfreich sein.
- Falls die Trident-Protokolle nicht genügend Informationen enthalten, können Sie versuchen, den Debug-Modus für Trident zu aktivieren, indem Sie das `-d` Flag an den Installationsparameter übergeben, abhängig von Ihrer Installationsoption.

  Bestätigen Sie dann, dass Debug gesetzt ist, indem Sie `./tridentctl logs -n trident` verwenden und im Protokoll nach `level=debug msg` suchen.

Installiert mit Operator

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

Dadurch werden alle Trident Pods neu gestartet, was einige Sekunden dauern kann. Sie können dies überprüfen, indem Sie die Spalte „AGE“ in der Ausgabe von `kubectl get pod -n trident` beobachten.

Für Trident 20.07 und 20.10 verwenden Sie `tprov` anstelle von `torc`.

Mit Helm installiert

```
helm upgrade <name> trident-operator-21.07.1-custom.tgz --set
tridentDebug=true`
```

Installiert mit tridentctl

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

- Sie können auch Debug-Logs für jedes Backend erhalten, indem Sie `debugTraceFlags` in Ihre Backend-Definition aufnehmen. Fügen Sie beispielsweise `debugTraceFlags: {"api":true, "method":true,}` hinzu, um API-Aufrufe und Methodendurchläufe in den Trident-Logs zu erhalten. Bestehende Backends können `debugTraceFlags` mit einem `tridentctl backend update` konfiguriert werden.
- Bei Verwendung von Red Hat Enterprise Linux CoreOS (RHCOS) muss sichergestellt werden, dass `iscsid` auf den Worker-Knoten aktiviert und standardmäßig gestartet ist. Dies kann durch die Verwendung von OpenShift MachineConfigs oder durch Anpassen der Ignition-Vorlagen erfolgen.
- Ein häufiges Problem, auf das Sie bei der Verwendung von Trident mit "Azure NetApp Files" stoßen könnten, ist, wenn die Mandanten- und Clientgeheimnisse aus einer App-Registrierung mit unzureichenden Berechtigungen stammen. Eine vollständige Liste der Trident-Anforderungen finden Sie unter "Azure NetApp Files"Konfiguration.
- Wenn es Probleme beim Einbinden eines PV in einen Container gibt, stellen Sie sicher, dass `rpcbind` installiert und ausgeführt wird. Verwenden Sie den erforderlichen Paketmanager für das Host-Betriebssystem und prüfen Sie, ob `rpcbind` läuft. Sie können den Status des `rpcbind` Dienstes überprüfen, indem Sie einen `systemctl status rpcbind` oder einen entsprechenden Befehl ausführen.
- Wenn ein Trident-Backend meldet, dass es sich im `failed` Status befindet, obwohl es zuvor funktioniert hat, liegt dies wahrscheinlich an einer Änderung der mit dem Backend verbundenen SVM-/Admin-Zugangsdaten. Das Aktualisieren der Backend-Informationen mit `tridentctl update backend` oder ein Neustart des Trident-Pods behebt dieses Problem.
- Falls bei der Installation von Trident mit Docker als Container-Laufzeitumgebung Berechtigungsprobleme auftreten, versuchen Sie die Installation von Trident mit dem `--in cluster=false` Flag. Dadurch wird kein Installer-Pod verwendet und Berechtigungsprobleme, die durch den `trident-installer` Benutzer verursacht werden, werden vermieden.
- Verwenden Sie das `uninstall parameter <Uninstalling Trident>` zur Bereinigung nach einem

fehlgeschlagenen Lauf. Standardmäßig entfernt das Skript die von Trident erstellten CRDs nicht, sodass eine Deinstallation und Neuinstallation auch in einer laufenden Bereitstellung sicher ist.

- Wenn Sie auf eine frühere Version von Trident downgraden möchten, führen Sie zuerst den `tridentctl uninstall` Befehl aus, um Trident zu entfernen. Laden Sie die gewünschte "Trident Version" herunter und installieren Sie sie mit dem `tridentctl install` Befehl.
- Nach einer erfolgreichen Installation, wenn eine PVC in der `Pending`-Phase feststeckt, kann das Ausführen von `kubectl describe pvc` zusätzliche Informationen darüber liefern, warum Trident es nicht geschafft hat, eine PV für diese PVC bereitzustellen.

Fehlgeschlagene Trident-Bereitstellung mit dem Operator

Wenn Sie Trident mithilfe des Operators bereitstellen, ändert sich der Status von `TridentOrchestrator Installing`zu `Installed`. Wenn Sie den `Failed` Status beobachten und der Operator sich nicht selbst wiederherstellen kann, sollten Sie die Protokolle des Operators mit folgendem Befehl überprüfen:

```
tridentctl logs -l trident-operator
```

Das Verfolgen der Logs des trident-operator-Containers kann darauf hinweisen, wo das Problem liegt. Ein solches Problem könnte beispielsweise darin bestehen, dass die erforderlichen Container-Images in einer abgeschotteten Umgebung nicht von Upstream-Registries abgerufen werden können.

Um zu verstehen, warum die Installation von Trident fehlgeschlagen ist, sollten Sie sich den `TridentOrchestrator` Status ansehen.

```
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 eine `TridentOrchestrator` existiert, die zur Installation von Trident verwendet wurde. Da jeder Kubernetes-Cluster nur eine Instanz von Trident haben kann, stellt der Operator sicher, dass zu jedem Zeitpunkt nur eine aktive `TridentOrchestrator` existiert, die er erstellen kann.

Darüber hinaus kann die Beobachtung des Zustands der Trident-Pods oft darauf hinweisen, ob etwas nicht stimmt.

```
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 deutlich sehen, dass die Pods nicht vollständig initialisiert werden können, weil ein oder mehrere Container-Images nicht abgerufen wurden.

Um das Problem zu beheben, sollten Sie die `TridentOrchestrator` CR bearbeiten. Alternativ können Sie `TridentOrchestrator` löschen und eine neue mit der geänderten und korrekten Definition erstellen.

Fehlgeschlagene Trident-Bereitstellung mit `tridentctl`

Um herauszufinden, was schiefgelaufen ist, können Sie das Installationsprogramm erneut mit dem `-d` Argument ausführen, das den Debug-Modus aktiviert und Ihnen hilft, das Problem zu verstehen:

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

Nachdem das Problem behoben wurde, können Sie die Installation wie folgt bereinigen und dann den `tridentctl install` Befehl erneut ausführen:

```
./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.
```

Trident und CRDs vollständig entfernen

Sie können Trident und alle erstellten CRDs und zugehörigen benutzerdefinierten Ressourcen vollständig entfernen.

WARNUNG	Dies kann nicht rückgängig gemacht werden. Führen Sie dies nur durch, wenn Sie eine komplett neue Installation von Trident wünschen. Um Trident zu deinstallieren, ohne die CRDs zu entfernen, siehe "Trident deinstallieren".

Trident Operator

So deinstallieren Sie Trident und entfernen CRDs vollständig mit dem Trident Operator:

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

Helm

So deinstallieren Sie Trident und entfernen CRDs vollständig mit Helm:

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

<code>tridentctl</code>

Um CRDs nach der Deinstallation von Trident vollständig zu entfernen, verwenden Sie `tridentctl`

```
tridentctl obliviate crd
```

NVMe-Knoten-Unstaging-Fehler mit RWX-Raw-Block-Namespaces auf Kubernetes 1.26

Wenn Sie Kubernetes 1.26 ausführen, kann das Unstaging von Knoten fehlschlagen, wenn NVMe/TCP mit RWX Raw Block Namespaces verwendet wird. Die folgenden Szenarien bieten eine Problemumgehung für den Fehler. Alternativ können Sie Kubernetes auf 1.27 aktualisieren.

Namespace und Pod gelöscht

Stellen Sie sich ein Szenario vor, in dem ein von Trident verwalteter Namespace (NVMe-Persistent Volume) an einen Pod angehängt ist. Wenn Sie den Namespace direkt vom ONTAP Backend löschen, bleibt der Unstaging-Prozess nach dem Versuch, den Pod zu löschen, hängen. Dieses Szenario hat keine Auswirkungen auf den Kubernetes-Cluster oder andere Funktionen.

Problemumgehung

Hängen Sie das persistente Volume (das diesem Namespace entspricht) vom jeweiligen Knoten aus und löschen Sie es.

Blockierte dataLIFs

```
 If you block (or bring down) all the dataLIFs of the NVMe Trident
backend, the unstaging process gets stuck when you attempt to delete the
pod. In this scenario, you cannot run any NVMe CLI commands on the
Kubernetes node.
.Problemumgehung
Aktivieren Sie die dataLIFS, um die volle Funktionalität
wiederherzustellen.
```

Gelöschte Namespace-Zuordnung

```
 If you remove the `hostNQN` of the worker node from the corresponding
subsystem, the unstaging process gets stuck when you attempt to delete the
pod. In this scenario, you cannot run any NVMe CLI commands on the
Kubernetes node.
.Problemumgehung
Fügen Sie das  `hostNQN` zurück zum Subsystem hinzu.
```

NFSv4.2-Clients melden „ungültiges Argument“ nach dem Upgrade von ONTAP, wenn sie erwarten, dass „v4.2-xattrs“ aktiviert ist

Nach dem Upgrade von ONTAP können NFSv4.2-Clients beim Versuch, NFSv4.2-Exporte einzubinden, „invalid argument“-Fehler melden. Dieses Problem tritt auf, wenn die `v4.2-xattrs` Option auf der SVM nicht aktiviert ist. Abhilfe: Aktivieren Sie die `v4.2-xattrs` Option auf der SVM oder aktualisieren Sie auf ONTAP 9.12.1 oder höher, wo diese Option standardmäßig aktiviert ist.

Support

NetApp bietet Unterstützung für Trident auf verschiedene Weise. Umfangreiche kostenlose Selbsthilfeoptionen stehen rund um die Uhr zur Verfügung, wie zum Beispiel Knowledgebase (KB)-Artikel und ein Discord-Kanal.

Trident Support-Lebenszyklus

Trident bietet drei Supportstufen basierend auf Ihrer Version. Siehe "NetApp Softwareversionsunterstützung für Definitionen".

Volle Unterstützung
: Trident bietet vollen Support für zwölf Monate ab dem Veröffentlichungsdatum.

Eingeschränkte Unterstützung
: Trident bietet nur eingeschränkten Support für die Monate 13 bis 24 ab dem Veröffentlichungsdatum.

Selbsthilfe
: Die Trident-Dokumentation ist für die Monate 25 bis 36 ab dem Veröffentlichungsdatum verfügbar.

Version	Volle Unterstützung	Eingeschränkte Unterstützung	Selbsthilfe
"26,02"	Februar 2026	Februar 2027	Februar 2028
"25,10"	Oktober 2026	Oktober 2027	Oktober 2028
"25,06"	Juni 2026	Juni 2027	Juni 2028
"25,02"	Februar 2026	Februar 2027	Februar 2028
"24,10"	—	Oktober 2026	Oktober 2027
"24,06"	—	Juni 2026	Juni 2027
"24,02"	—	Februar 2026	Februar 2027
"23,10"	—	—	Oktober 2026
"23,07"	—	—	Juli 2026
"23,04"	—	—	April 2026

Selbsthilfe

Eine vollständige Liste der Artikel zur Fehlerbehebung finden Sie unter "NetApp Knowledgebase (Anmeldung erforderlich)".

Community-Support

Es gibt eine lebendige öffentliche Community von Container-Nutzern (einschließlich Trident-Entwicklern) auf unserer "Discord-Kanal". Dies ist ein großartiger Ort, um allgemeine Fragen zum Projekt zu stellen und verwandte Themen mit Gleichgesinnten zu diskutieren.

NetApp technische Unterstützung

Um Hilfe mit Trident zu erhalten, erstellen Sie ein Support-Bundle mit `tridentctl logs -a -n trident` und senden Sie es an `NetApp Support <Getting Help>`.

Für weitere Informationen

- "Trident-Ressourcen"
- "Kubernetes Hub"

Referenz

Trident-Ports

Erfahren Sie mehr über die Ports, die Trident für die Kommunikation verwendet.

Überblick

Trident nutzt verschiedene Ports für die Kommunikation innerhalb von Kubernetes-Clustern und mit Storage-Backends. Im Folgenden finden Sie eine Zusammenfassung der wichtigsten Ports, ihrer Zwecke und Sicherheitsaspekte.

- **Ausgehender Fokus**: Kubernetes-Knoten (Controller und Worker) initiieren primär Datenverkehr zu Storage-LIFs/IPs, daher sollten iptables-Regeln ausgehenden Datenverkehr von Knoten-IPs zu bestimmten Storage-IPs auf diesen Ports zulassen. Vermeiden Sie allgemeine „Beliebig-zu-Beliebig“-Regeln.
- **Eingehende Beschränkungen**: Beschränken Sie interne Trident-Ports auf clusterinternen Datenverkehr (zum Beispiel mit CNI wie Calico). Keine unnötige eingehende Exponierung auf den Host-Firewalls.
- **Protokollsicherheit**:
  - Verwenden Sie nach Möglichkeit TCP (zuverlässiger).
  - Aktivieren Sie CHAP/IPsec für iSCSI, falls sensibel; TLS/HTTPS für das Management (Port 443/8443).
  - Für NFSv4 (Standard in Trident) entfernen Sie UDP-/ältere NFSv3-Ports (zum Beispiel 4045-4049), wenn sie nicht benötigt werden.
  - Beschränken Sie auf vertrauenswürdige Subnetze; überwachen Sie mit Tools wie Prometheus (optional Port 8001).

Anschlüsse für Controller-Knoten

Diese Ports sind primär für den Trident operator (Backend-Management) vorgesehen. Alle internen Ports sind auf Pod-Ebene; erlauben Sie sie auf Knoten nur, wenn die Host-Firewall mit CNI interferiert.

Port/Protokoll	Richtung	Zweck	Treiber/Protokoll	Sicherheitshinweise
TCP 8000	Eingehend/Ausgehend (cluster-intern)	Trident REST server (Operator-Controller-Kommunikation)	Alle	Beschränken Sie auf Pod-CIDRs; keine externe Exposition.
TCP 8443	Eingehend/Ausgehend (cluster-intern)	Backchannel HTTPS (sichere interne API)	Alle	TLS-verschlüsselt; auf Kubernetes Service Mesh beschränken, falls verwendet.
TCP 8001	Eingehend (clusterintern, optional)	Prometheus-Metriken	Alle	Nur für Überwachungstools (zum Beispiel mit RBAC) zugänglich machen; deaktivieren, wenn nicht verwendet.
TCP 443	Ausgehend	HTTPS zu ONTAP SVM/Cluster-Mgmt LIF	ONTAP (alle), ANF	TLS-Zertifikatvalidierung erforderlich; nur auf mgmt LIF-IPs beschränken.

Port/Protokoll	Richtung	Zweck	Treiber/Protokoll	Sicherheitshinweise
TCP 8443	Ausgehend	HTTPS-zu-E-Series-Web Services Proxy	E-Series (iSCSI)	Standardmäßige REST API; Verwendung von Zertifikaten; konfigurierbar im Backend-YAML.

Ports für Worker-Knoten

Diese Ports sind für CSI-Knoten-Daemonsets und Pod-Mounts vorgesehen. Datenports sind ausgehende Verbindungen zu Storage Data LIFs; fügen Sie NFSv3-Extras hinzu, wenn Sie NFSv3 verwenden (optional für NFSv4).

Port/Protokoll	Richtung	Zweck	Treiber/Protokoll	Sicherheitshinweise
TCP 17546	Eingehend (lokal zum Pod)	CSI-Knoten-Liveness/Readiness-Probes	Alle	Konfigurierbar (--probe-port); sicherstellen, dass keine Host-Konflikte bestehen; nur lokal.
TCP 8000	Eingehend/Ausgehend (cluster-intern)	Trident REST-Server	Alle	Wie oben; pod-internal.
TCP 8443	Eingehend/Ausgehend (cluster-intern)	Backchannel HTTPS	Alle	Wie oben.
TCP 8001	Eingehend (clusterintern, optional)	Prometheus-Metriken	Alle	Wie oben.
TCP 443	Ausgehend	HTTPS zu ONTAP SVM/Cluster-Mgmt LIF	ONTAP (alle), ANF	Wie oben; wird zur Ermittlung verwendet.
TCP 8443	Ausgehend	HTTPS-zu-E-Series-Web Services Proxy	E-Series (iSCSI)	Wie oben.
TCP/UDP 111	Ausgehend	RPCBIND/portmapper	ONTAP-NAS (NFSv3/v4), ANF (NFS)	Erforderlich für v3; optional für v4 (firewall offload); einschränken, wenn ausschließlich NFSv4 verwendet wird.
TCP/UDP 2049	Ausgehend	NFS-Daemon	ONTAP-NAS (NFSv3/v4), ANF (NFS)	Kerndaten; bekannt; TCP für Zuverlässigkeit verwenden.
TCP/UDP 635	Ausgehend	Mount-Daemon	ONTAP-NAS (NFSv3/v4), ANF (NFS)	Montage; bidirektionale Rückrufe möglich (bei Bedarf eingehende ephemere Verbindungen zulassen).
UDP 4045	Ausgehend	NFS-Sperrmanager (nlockmgr)	ONTAP-NAS (NFSv3)	Dateisperrung; für v4 überspringen (pNFS handles); nur UDP.

Port/Protokoll	Richtung	Zweck	Treiber/Protokoll	Sicherheitshinweise
UDP 4046	Ausgehend	NFS-Statusmonitor (statd)	ONTAP-NAS (NFSv3)	Benachrichtigungen; möglicherweise werden eingehende ephemere Ports (1024-65535) für Rückrufe benötigt.
UDP 4049	Ausgehend	NFS-Quota-Daemon (rquotad)	ONTAP-NAS (NFSv3)	Kontingente; für v4 überspringen.
TCP 3260	Ausgehend	iSCSI-Ziel (Erkennung/Daten/CHAP)	ONTAP-SAN (iSCSI), E-Series (iSCSI)	Bekannt; CHAP-Authentifizierung über diesen Port; gegenseitiges CHAP zur Sicherheit aktivieren.
TCP 445	Ausgehend	SMB/CIFS	ONTAP-NAS (SMB), ANF (SMB)	Bekannt; verwenden Sie SMB3 mit Verschlüsselung (Trident annotation netapp.io/smb-encryption=true).
TCP/UDP 88 (optional)	Ausgehend	Kerberos-Authentifizierung	ONTAP (NFS/SMB/iSCSI mit Kerb)	Bei Verwendung von Kerberos (nicht Standardeinstellung); zu AD-Servern, nicht zum Storage.
TCP/UDP 389 (optional)	Ausgehend	LDAP	ONTAP (NFS/SMB mit LDAP)	Ähnlich; für Namensauflösung/Authentifizierung; auf AD beschränken.

HINWEIS Der Port für die Liveness-/Readiness-Prüfung kann während der Installation mit dem `--probe-port`-Flag geändert werden. Es ist wichtig sicherzustellen, dass dieser Port nicht von einem anderen Prozess auf den Worker-Knoten verwendet wird.

Trident REST API

Während "tridentctl commands und Optionen" die einfachste Möglichkeit sind, mit der Trident REST API zu interagieren, können Sie den REST-Endpunkt auch direkt verwenden, wenn Sie dies bevorzugen.

Wann sollte die REST API verwendet werden?

REST API ist nützlich für fortgeschrittene Installationen, die Trident als eigenständige Binärdatei in Nicht-Kubernetes-Bereitstellungen verwenden.

Für bessere Sicherheit ist Trident `REST API` standardmäßig auf localhost beschränkt, wenn es in einem Pod ausgeführt wird. Um dieses Verhalten zu ändern, müssen Sie das Argument von Trident `-address` in seiner Pod-Konfiguration festlegen.

Verwendung der REST API

Beispiele dafür, wie diese APIs aufgerufen werden, erhalten Sie, wenn Sie das Debug-(`-d`-Flag übergeben. Weitere Informationen finden sich unter "Trident mit tridentctl verwalten".

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>`

Ruft die Details des benannten Objekts ab.

POST

`POST <trident-address>/trident/v1/<object-type>`

Erzeugt ein Objekt des angegebenen Typs.

- Für die Erstellung des Objekts ist eine JSON-Konfiguration erforderlich. Die Spezifikation der einzelnen Objekttypen finden Sie unter "Trident mit tridentctl verwalten".
- Wenn das Objekt bereits existiert, variiert das Verhalten: Backends aktualisieren das bestehende Objekt, während bei allen anderen Objekttypen der Vorgang fehlschlägt.

LÖSCHEN

`DELETE <trident-address>/trident/v1/<object-type>/<object-name>`

Löscht die benannte Ressource.

HINWEIS	Zu Backends oder Speicherklassen gehörende Volumes bleiben bestehen; diese müssen separat gelöscht werden. Weitere Informationen finden sich unter "Trident mit tridentctl verwalten".

Befehlszeilenoptionen

Trident stellt mehrere Befehlszeilenoptionen für den Trident Orchestrator bereit. Mit diesen Optionen können Sie Ihre Bereitstellung anpassen.

Protokollierung

`-debug`

Aktiviert die Debugging-Ausgabe.

`-loglevel <level>`

Legt das Protokollierungslevel (debug, info, warn, error, fatal) fest. Standardmäßig auf info gesetzt.

Kubernetes

-k8s_pod

Verwenden Sie diese Option oder `-k8s_api_server`, um die Kubernetes-Unterstützung zu aktivieren. Wenn Sie diese Einstellung festlegen, verwendet Trident die Kubernetes-Dienstkontoinformationen seines eigenen Pods, um den API-Server zu kontaktieren. Dies funktioniert nur, wenn Trident als Pod in einem Kubernetes-Cluster mit aktivierten Dienstkonten ausgeführt wird.

-k8s_api_server <insecure-address:insecure-port>

Verwenden Sie diese Option oder `-k8s_pod`, um die Kubernetes-Unterstützung zu aktivieren. Wenn angegeben, verbindet sich Trident mit dem Kubernetes-API-Server über die bereitgestellte unsichere Adresse und den Port. Dadurch kann Trident außerhalb eines Pods bereitgestellt werden; es werden jedoch nur unsichere Verbindungen zum API-Server unterstützt. Um eine sichere Verbindung herzustellen, stellen Sie Trident in einem Pod mit der `-k8s_pod` Option bereit.

Docker

-volume_driver <name>

Name des Treibers, der bei der Registrierung des Docker-Plugins verwendet wird. Standardmäßig `netapp`.

-driver_port <port-number>

Lauschen Sie auf diesem Port statt auf einem UNIX-Domain-Socket.

-config <file>

Erforderlich; Sie müssen diesen Pfad zu einer Backend-Konfigurationsdatei angeben.

REST

-address <ip-or-host>

Gibt die Adresse an, auf der der REST-Server von Trident lauschen soll. Standardmäßig ist dies localhost. Wenn auf localhost gelauscht wird und Trident in einem Kubernetes-Pod ausgeführt wird, ist die REST-Schnittstelle von außerhalb des Pods nicht direkt zugänglich. Verwenden Sie `-address ""`, um die REST-Schnittstelle von der Pod-IP-Adresse aus zugänglich zu machen.

ACHTUNG	Trident REST interface kann so konfiguriert werden, dass sie nur unter 127.0.0.1 (für IPv4) oder [::1] (für IPv6) lauscht und Anfragen beantwortet.

-port <port-number>

Gibt den Port an, auf dem der REST-Server von Trident lauschen soll. Standardmäßig 8000.

-rest

Aktiviert die REST-Schnittstelle. Standardmäßig auf true gesetzt.

Kubernetes- und Trident-Objekte

Sie können mit Kubernetes und Trident über REST-APIs interagieren, indem Sie Ressourcenobjekte lesen und schreiben. Es gibt verschiedene Ressourcenobjekte, die die Beziehungen zwischen Kubernetes und Trident, Trident und Speicher sowie Kubernetes und Speicher bestimmen. Einige dieser Objekte werden über Kubernetes

und die anderen über Trident verwaltet.

Wie interagieren die Objekte miteinander?

Vielleicht ist der einfachste Weg, die Objekte, ihren Zweck und ihre Interaktion zu verstehen, einer einzelnen Speicheranfrage eines Kubernetes-Benutzers zu folgen:

1. Ein Benutzer erstellt eine `PersistentVolumeClaim` Anfrage für eine neue `PersistentVolume` einer bestimmten Größe von einem Kubernetes `StorageClass`, der zuvor vom Administrator konfiguriert wurde.
2. Kubernetes `StorageClass` identifiziert Trident als seinen Provisioner und enthält Parameter, die Trident angeben, wie ein Volume für die angeforderte Klasse bereitgestellt werden soll.
3. Trident betrachtet seine eigenen `StorageClass` mit demselben Namen, der die passende `Backends` und `StoragePools` identifiziert, die es zur Bereitstellung von Volumes für die Klasse verwenden kann.
4. Trident stellt Speicher auf einem passenden Backend bereit und erstellt zwei Objekte: ein `PersistentVolume` in Kubernetes, das Kubernetes mitteilt, wie das Volume gefunden, eingebunden und behandelt werden soll, und ein Volume in Trident, das die Beziehung zwischen dem `PersistentVolume` und dem eigentlichen Speicher aufrechterhält.
5. Kubernetes bindet die `PersistentVolumeClaim` an die neue `PersistentVolume`. Pods, die die `PersistentVolumeClaim` Mount-Anweisung enthalten, mounten dieses PersistentVolume auf jedem Host, auf dem sie ausgeführt werden.
6. Ein Benutzer erstellt eine `VolumeSnapshot` einer bestehenden PVC, indem er eine `VolumeSnapshotClass` verwendet, die auf Trident zeigt.
7. Trident identifiziert das dem PVC zugeordnete Volume und erstellt einen Snapshot dieses Volumes im Backend. Es erstellt außerdem eine `VolumeSnapshotContent`, die Kubernetes anweist, wie der Snapshot identifiziert werden kann.
8. Ein Benutzer kann eine `PersistentVolumeClaim` mit `VolumeSnapshot` als Quelle erstellen.
9. Trident identifiziert den erforderlichen Snapshot und führt die gleichen Schritte aus, die auch beim Erstellen eines `PersistentVolume` und eines `Volume` erforderlich sind.

TIPP Für weiterführende Informationen zu Kubernetes-Objekten empfehlen wir Ihnen dringend, den "Persistente Volumes" Abschnitt der Kubernetes-Dokumentation zu lesen.

Kubernetes `PersistentVolumeClaim`-Objekte

Ein Kubernetes `PersistentVolumeClaim`-Objekt ist eine Speicheranforderung, die von einem Kubernetes-Cluster-Benutzer gestellt wird.

Zusätzlich zur Standardspezifikation ermöglicht Trident den Benutzern, die folgenden volumespezifischen Annotationen anzugeben, wenn sie die in der Backend-Konfiguration festgelegten Standardeinstellungen überschreiben möchten:

Anmerkung	Volume-Option	Unterstützte Treiber
trident.netapp.io/fileSystem	fileSystem	ontap-san, solidfire-san,ontap-san-economy

Anmerkung	Volume-Option	Unterstützte Treiber
trident.netapp.io/cloneFromPVC	cloneSourceVolume	ontap-nas, ontap-san, solidfire-san, azure-netapp-files, ontap-san-economy
trident.netapp.io/splitOnClone	splitOnClone	ontap-nas, ontap-san
trident.netapp.io/protocol	Protokoll	beliebig
trident.netapp.io/exportPolicy	exportPolicy	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
trident.netapp.io/snapshotDirectory	snapshotDirectory	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup
trident.netapp.io/unixPermissions	unixPermissions	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup
trident.netapp.io/blockSize	blockSize	solidfire-san
trident.netapp.io/skipRecoveryQueue	skipRecoveryQueue	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, ontap-san-economy

Wenn das erstellte PV die `Delete` Reclaim-Richtlinie hat, löscht Trident sowohl das PV als auch das zugehörige Volume, sobald das PV freigegeben wird (das heißt, wenn der Benutzer das PVC löscht). Schlägt der Löschvorgang fehl, markiert Trident das PV entsprechend und wiederholt die Operation regelmäßig, bis sie erfolgreich ist oder das PV manuell gelöscht wird. Verwendet das PV die `Retain` Richtlinie, ignoriert Trident es und geht davon aus, dass der Administrator es aus Kubernetes und dem Backend entfernt, sodass das Volume vor seiner Entfernung gesichert oder überprüft werden kann. Beachten Sie, dass das Löschen des PV nicht dazu führt, dass Trident das zugehörige Volume löscht. Sie sollten es über die REST-API entfernen (`tridentctl`.

Trident unterstützt die Erstellung von Volume-Snapshots gemäß der CSI-Spezifikation: Sie können einen Volume-Snapshot erstellen und ihn als Datenquelle zum Klonen vorhandener PVCs verwenden. Auf diese Weise können zeitpunktgenaue Kopien von PVs in Form von Snapshots für Kubernetes bereitgestellt werden. Die Snapshots können dann zum Erstellen neuer PVs verwendet werden. Sehen Sie sich `On-Demand Volume Snapshots` an, um zu sehen, wie das funktioniert.

Trident bietet außerdem die `cloneFromPVC` und `splitOnClone` Annotationen zum Erstellen von Klonen. Mit diesen Annotationen können Sie ein PVC klonen, ohne die CSI-Implementierung verwenden zu müssen.

Hier ist ein Beispiel: Wenn ein Benutzer bereits eine PVC namens `mysql` besitzt, kann der Benutzer eine neue PVC namens `mysqlclone` erstellen, indem er die Annotation wie `trident.netapp.io/cloneFromPVC: mysql` verwendet. Mit dieser Annotation klont Trident das Volume, das der mysql PVC entspricht, anstatt ein Volume von Grund auf bereitzustellen.

Beachten Sie folgende Punkte:

- NetApp empfiehlt das Klonen eines ungenutzten Volumes.

- Ein PVC und sein Klon sollten sich im selben Kubernetes-Namespace befinden und die gleiche Storage Class haben.
- Mit den `ontap-nas` und `ontap-san` Treibern kann es wünschenswert sein, die PVC-Annotation `trident.netapp.io/splitOnClone` in Verbindung mit `trident.netapp.io/cloneFromPVC` festzulegen. Wenn `trident.netapp.io/splitOnClone` auf `true` gesetzt ist, trennt Trident das geklonte Volume vom übergeordneten Volume und entkoppelt so den Lebenszyklus des geklonten Volumes vollständig vom übergeordneten Volume, allerdings auf Kosten eines geringeren Speicherwirkungsgrads. Wenn `trident.netapp.io/splitOnClone` nicht gesetzt ist oder auf `false` gesetzt wird, führt dies zu einem geringeren Speicherplatzverbrauch im Backend, allerdings entstehen dadurch Abhängigkeiten zwischen dem übergeordneten und dem geklonten Volume, sodass das übergeordnete Volume erst gelöscht werden kann, wenn der Klon zuvor gelöscht wurde. Ein Szenario, in dem das Aufteilen des Klons sinnvoll ist, ist das Klonen eines leeren Datenbank-Volumes, bei dem zu erwarten ist, dass sich Volume und Klon stark unterscheiden und nicht von den Speichereffizienzen profitieren, die ONTAP bietet.

Das `sample-input` Verzeichnis enthält Beispiele für PVC-Definitionen zur Verwendung mit Trident. Siehe für eine vollständige Beschreibung der Parameter und Einstellungen, die mit Trident-Volumes verbunden sind.

Kubernetes `PersistentVolume`-Objekte

Ein Kubernetes `PersistentVolume`-Objekt repräsentiert einen Speicherbereich, der dem Kubernetes-Cluster zur Verfügung gestellt wird. Es hat einen Lebenszyklus, der unabhängig von dem Pod ist, der ihn verwendet.

HINWEIS	Trident erstellt `PersistentVolume` Objekte und registriert sie automatisch beim Kubernetes-Cluster basierend auf den Volumes, die es bereitstellt. Sie müssen diese nicht selbst verwalten.

Wenn Sie eine PVC erstellen, die auf ein Trident-basiertes `StorageClass` verweist, stellt Trident ein neues Volume mit der entsprechenden Speicherklasse bereit und registriert ein neues PV für dieses Volume. Bei der Konfiguration des bereitgestellten Volumes und des zugehörigen PV befolgt Trident die folgenden Regeln:

- Trident generiert einen PV-Namen für Kubernetes und einen internen Namen, den es zur Bereitstellung des Speichers verwendet. In beiden Fällen stellt es sicher, dass die Namen innerhalb ihres Geltungsbereichs eindeutig sind.
- Die Größe des Volumens entspricht so genau wie möglich der im PVC angeforderten Größe, kann jedoch je nach Plattform auf die nächstzuweisbare Menge aufgerundet werden.

Kubernetes `StorageClass`-Objekte

Kubernetes `StorageClass`-Objekte werden anhand ihres Namens in `PersistentVolumeClaims` spezifiziert, um Speicher mit einer Reihe von Eigenschaften bereitzustellen. Die Speicherklasse selbst identifiziert den zu verwendenden Provisionierer und definiert diese Eigenschaften in einer für den Provisionierer verständlichen Weise.

Es handelt sich um eines von zwei grundlegenden Objekten, die vom Administrator erstellt und verwaltet werden müssen. Das andere ist das Trident Backend-Objekt.

Ein Kubernetes `StorageClass`-Objekt, das Trident verwendet, sieht folgendermaßen 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 geben Trident an, wie Volumes für die Klasse bereitgestellt werden sollen.

Die Parameter der Speicherklasse sind:

Attribut	Typ	Erforderlich	Beschreibung
Attribute	map[string]string	nein	Siehe den Abschnitt Attribute unten
storagePools	map[string]StringList	nein	Zuordnung von Backend-Namen zu Listen von Speicherpools innerhalb
additionalStoragePools	map[string]StringList	nein	Zuordnung von Backend-Namen zu Listen von Speicherpools innerhalb
excludeStoragePools	map[string]StringList	nein	Zuordnung von Backend-Namen zu Listen von Speicherpools innerhalb

Speicherattribute und ihre möglichen Werte können in Attribute zur Auswahl von Speicherpools und Kubernetes-Attribute eingeteilt werden.

Auswahlattribute für Speicherpools

Diese Parameter legen fest, welche von Trident verwalteten Speicherpools zur Bereitstellung von Volumes eines bestimmten Typs verwendet werden sollen.

Attribut	Typ	Werte	Angebot	Anfrage	Unterstützt von
Medien[1]	Zeichenkette	hdd, hybrid, ssd	Pool enthält Medien dieses Typs; hybrid bedeutet beides	Medientyp angegeben	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, solidfire-san
provisioningType	Zeichenkette	dünn, dick	Pool unterstützt diese Bereitstellungsmethode	Bereitstellungsmethode angegeben	dick: alle ontap; dünn: alle ontap & solidfire-san

Attribut	Typ	Werte	Angebot	Anfrage	Unterstützt von
backendType	Zeichenkette	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, solidfire-san, azure-netapp-files, ontap-san-economy	Pool gehört zu dieser Art von Backend	Backend angegeben	Alle Treiber
Momentaufnahmen	bool	wahr, falsch	Pool unterstützt Volumes mit Snapshots	Volume mit aktivierten Snapshots	ontap-nas, ontap-san, solidfire-san
Klone	bool	wahr, falsch	Pool unterstützt das Klonen von Volumes	Volume mit aktivierten Klonen	ontap-nas, ontap-san, solidfire-san
Verschlüsselung	bool	wahr, falsch	Pool unterstützt verschlüsselte Volumes	Volume mit aktivierter Verschlüsselung	ontap-nas, ontap-nas-economy, ontap-nas-flexgroups, ontap-san
IOPS	int	positive ganze Zahl	Pool ist in der Lage, IOPS in diesem Bereich zu garantieren	Volume garantiert diese IOPS	solidfire-san

[1]: Wird von ONTAP Select-Systemen nicht unterstützt

In den meisten Fällen beeinflussen die angeforderten Werte die Bereitstellung direkt; beispielsweise führt die Anforderung von Thick Provisioning zu einem Thick-Provisioning-Volume. Ein Element-Speicherpool verwendet jedoch seine angebotenen minimalen und maximalen IOPS-Werte zur Festlegung der QoS-Werte, anstatt des angeforderten Wertes. In diesem Fall wird der angeforderte Wert nur zur Auswahl des Speicherpools verwendet.

Im Idealfall können Sie `attributes` allein verwenden, um die Eigenschaften des Speichers zu modellieren, die Sie benötigen, um die Anforderungen einer bestimmten Klasse zu erfüllen. Trident erkennt und wählt automatisch Speicherpools aus, die *allen* `attributes` entsprechen, die Sie angeben.

Falls Sie nicht in der Lage sind, `attributes` die richtigen Pools für eine Klasse automatisch auszuwählen, können Sie die `storagePools` und `additionalStoragePools` Parameter verwenden, um die Pools weiter einzugrenzen oder sogar eine bestimmte Gruppe von Pools auszuwählen.

Sie können den `storagePools` Parameter verwenden, um die Menge der Pools, die mit einem angegebenen `attributes` übereinstimmen, weiter einzuschränken. Anders ausgedrückt: Trident verwendet für die Bereitstellung die Schnittmenge der Pools, die durch die `attributes` und `storagePools` Parameter identifiziert werden. Sie können entweder einen der Parameter allein oder beide zusammen verwenden.

Sie können den `additionalStoragePools` Parameter verwenden, um die Menge der Pools zu erweitern, die Trident für die Bereitstellung verwendet, unabhängig von den durch die `attributes` und `storagePools` Parameter ausgewählten Pools.

Sie können den `excludeStoragePools` Parameter verwenden, um die von Trident für die Bereitstellung verwendeten Pools zu filtern. Durch die Verwendung dieses Parameters werden alle übereinstimmenden Pools entfernt.

In den `storagePools` und `additionalStoragePools` Parametern hat jeder Eintrag die Form `<backend>:<storagePoolList>`, wobei `<storagePoolList>` eine durch Kommas getrennte Liste von Speicherpools für das angegebene Backend ist. Ein Wert für `additionalStoragePools` könnte beispielsweise so 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 die Listenwerte. Sie können `tridentctl get backend` verwenden, um die Liste der Backends und ihrer Pools zu erhalten.

Kubernetes-Attribute

Diese Attribute haben keinen Einfluss auf die Auswahl von Speicherpools/Backends durch Trident während der dynamischen Bereitstellung. Stattdessen liefern diese Attribute einfach Parameter, die von Kubernetes Persistent Volumes unterstützt werden. Worker-Knoten sind für Dateisystem-Erstellungsvorgänge verantwortlich und benötigen möglicherweise Dateisystem-Dienstprogramme wie xfsprogs.

Attribut	Typ	Werte	Beschreibung	Relevante Treiber	Kubernetes-Version
fsType	Zeichenkette	ext4, ext3, xfs	Der Dateisystemtyp für Blockvolumes	solidfire-san, ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, ontap-san-economy	Alle
allowVolumeExpansion	boolescher Wert	wahr, falsch	Aktivieren oder Deaktivieren der Unterstützung für das Vergrößern der PVC-Größe	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, ontap-san-economy, solidfire-san, azure-netapp-files	1.11+
volumeBindingMode	Zeichenkette	Sofort, WaitForFirstConsumer	Wählen Sie aus, wann die Volumenbindung und die dynamische Bereitstellung erfolgt.	Alle	1.19 - 1.26

TIPP

- Der `fsType` Parameter wird verwendet, um den gewünschten Dateisystemtyp für SAN-LUNs zu steuern. Außerdem verwendet Kubernetes das Vorhandensein von `fsType` in einer Storage-Klasse, um anzuzeigen, dass ein Dateisystem existiert. Die Volume-Eigentümerschaft kann mit dem `fsGroup` Sicherheitskontext eines Pods nur gesteuert werden, wenn `fsType` gesetzt ist. Siehe "Kubernetes: Konfigurieren eines Security Context für einen Pod oder Container" für einen Überblick zur Festlegung der Volume-Eigentümerschaft mithilfe des `fsGroup` Kontexts. Kubernetes wendet den `fsGroup` Wert nur an, wenn:
  - `fsType` wird in der Speicherklasse festgelegt.
  - Der PVC-Zugriffsmodus ist RWO.

  Bei NFS-Speichertreibern ist ein Dateisystem bereits als Teil des NFS-Exports vorhanden. Um `fsGroup` die Speicherklasse zu verwenden, muss dennoch ein `fsType` angegeben werden. Sie können diesen auf `nfs` oder einen beliebigen Wert ungleich null setzen.
- Weitere Einzelheiten zur Volumenerweiterung finden Sie unter "Volumen erweitern".
- Das Trident-Installationspaket enthält mehrere Beispieldefinitionen für Speicherklassen zur Verwendung mit Trident in `sample-input/storage-class-*.yaml`. Das Löschen einer Kubernetes-Speicherklasse führt dazu, dass die entsprechende Trident-Speicherklasse ebenfalls gelöscht wird.

Kubernetes `VolumeSnapshotClass`-Objekte

Kubernetes `VolumeSnapshotClass`-Objekte sind analog zu `StorageClasses`. Sie helfen dabei, mehrere Speicherklassen zu definieren und werden von Volume-Snapshots referenziert, um den Snapshot mit der erforderlichen Snapshot-Klasse zu verknüpfen. Jeder Volume-Snapshot ist mit einer einzigen Volume-Snapshot-Klasse verknüpft.

A `VolumeSnapshotClass` sollte von einem Administrator definiert werden, um Snapshots zu erstellen. Eine Volume-Snapshot-Klasse wird mit der folgenden Definition erstellt:

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

Das `driver` gibt Kubernetes an, dass Anfragen für Volume-Snapshots der `csi-snapclass` Klasse von Trident verarbeitet werden. Das `deletionPolicy` gibt die Aktion an, die ausgeführt werden soll, wenn ein Snapshot gelöscht werden muss. Wenn `deletionPolicy` auf `Delete` gesetzt ist, werden sowohl die Volume-Snapshot-Objekte als auch der zugrunde liegende Snapshot auf dem Storage-Cluster entfernt, wenn ein Snapshot gelöscht wird. Alternativ bedeutet das Setzen auf `Retain`, dass `VolumeSnapshotContent` und der physische Snapshot erhalten bleiben.

Kubernetes `VolumeSnapshot`-Objekte

Ein Kubernetes `VolumeSnapshot` Objekt ist eine Anfrage zum Erstellen eines Snapshots eines Volumes.

Genau wie ein PVC eine Anfrage eines Benutzers für ein Volume darstellt, ist ein Volume Snapshot eine Anfrage eines Benutzers zum Erstellen eines Snapshots eines bestehenden PVC.

Wenn eine Volume-Snapshot-Anfrage eingeht, verwaltet Trident automatisch die Erstellung des Snapshots für das Volume im Backend und stellt den Snapshot durch die Erstellung eines eindeutigen `VolumeSnapshotContent` Objekts bereit. Sie können Snapshots von bestehenden PVCs erstellen und die Snapshots als DataSource beim Erstellen neuer PVCs verwenden.

HINWEIS	Der Lebenszyklus eines VolumeSnapshot ist unabhängig vom Quell-PVC: Ein Snapshot bleibt auch nach dem Löschen des Quell-PVC erhalten. Beim Löschen eines PVC mit zugehörigen Snapshots markiert Trident das zugehörige Datenträgervolumen im Status **Deleting**, entfernt es aber nicht vollständig. Das Volume wird entfernt, wenn alle zugehörigen Snapshots gelöscht sind.

Kubernetes `VolumeSnapshotContent`-Objekte

Ein Kubernetes `VolumeSnapshotContent`-Objekt stellt einen Snapshot eines bereits bereitgestellten Volumes dar. Es ist analog zu einem `PersistentVolume` und kennzeichnet einen bereitgestellten Snapshot im Speichercluster. Ähnlich wie `PersistentVolumeClaim` und `PersistentVolume` Objekten, wenn ein Snapshot erstellt wird, verwaltet das `VolumeSnapshotContent` Objekt eine Eins-zu-Eins-Zuordnung zu dem `VolumeSnapshot` Objekt, das die Snapshot-Erstellung angefordert hat.

Das `VolumeSnapshotContent` Objekt enthält Details, die den Snapshot eindeutig identifizieren, wie zum Beispiel das `snapshotHandle`. Dies `snapshotHandle` ist eine eindeutige Kombination aus dem Namen des PV und dem Namen des `VolumeSnapshotContent` Objekts.

Wenn eine Snapshot-Anfrage eingeht, erstellt Trident den Snapshot im Backend. Nachdem der Snapshot erstellt wurde, konfiguriert Trident ein `VolumeSnapshotContent` Objekt und stellt den Snapshot somit der Kubernetes-API zur Verfügung.

HINWEIS	Normalerweise müssen Sie das `VolumeSnapshotContent` Objekt nicht verwalten. Eine Ausnahme besteht, wenn Sie ein "einen Volume-Snapshot importieren" außerhalb von Trident erstellt haben.

Kubernetes `VolumeGroupSnapshotClass`-Objekte

Kubernetes `VolumeGroupSnapshotClass`-Objekte sind analog zu `VolumeSnapshotClass`. Sie helfen dabei, mehrere Speicherklassen zu definieren und werden von Volume-Gruppen-Snapshots referenziert, um den Snapshot mit der erforderlichen Snapshot-Klasse zu verknüpfen. Jeder Volume-Gruppen-Snapshot ist mit genau einer Volume-Gruppen-Snapshot-Klasse verknüpft.

A `VolumeGroupSnapshotClass` sollte von einem Administrator definiert werden, um eine Gruppe von Snapshots zu erstellen. Eine Volume Group Snapshot Class wird mit der folgenden Definition erstellt:

```
apiVersion: groupsnapshot.storage.k8s.io/v1beta1
kind: VolumeGroupSnapshotClass
metadata:
  name: csi-group-snap-class
  annotations:
    kubernetes.io/description: "Trident group snapshot class"
driver: csi.trident.netapp.io
deletionPolicy: Delete
```

Die `driver` gibt in Kubernetes an, dass Anfragen für Volume-Gruppen-Snapshots der `csi-group-snap-class` Klasse von Trident verarbeitet werden. Die `deletionPolicy` gibt die Aktion an, die ausgeführt werden soll, wenn ein Gruppen-Snapshot gelöscht werden muss. Wenn `deletionPolicy` auf `Delete` gesetzt ist, werden sowohl die Volume-Gruppen-Snapshot-Objekte als auch der zugrunde liegende Snapshot auf dem Storage-Cluster entfernt, wenn ein Snapshot gelöscht wird. Alternativ bedeutet das Setzen auf `Retain`, dass `VolumeGroupSnapshotContent` und der physische Snapshot erhalten bleiben.

Kubernetes `VolumeGroupSnapshot`-Objekte

Ein Kubernetes `VolumeGroupSnapshot`-Objekt ist eine Anfrage zum Erstellen eines Snapshots mehrerer Volumes. Genau wie ein PVC eine Anfrage eines Benutzers für ein Volume darstellt, ist ein Volume-Gruppen-Snapshot eine Anfrage eines Benutzers zum Erstellen eines Snapshots eines bestehenden PVC.

Wenn eine Volume-Gruppen-Snapshot-Anfrage eingeht, verwaltet Trident automatisch die Erstellung des Gruppen-Snapshots für die Volumes im Backend und stellt den Snapshot durch die Erstellung eines eindeutigen `VolumeGroupSnapshotContent` Objekts bereit. Sie können Snapshots von bestehenden PVCs erstellen und die Snapshots als DataSource beim Erstellen neuer PVCs verwenden.

HINWEIS	Der Lebenszyklus eines VolumeGroupSnapshot ist unabhängig vom Quell-PVC: Ein Snapshot bleibt auch nach dem Löschen des Quell-PVC erhalten. Beim Löschen eines PVC mit zugehörigen Snapshots markiert Trident das zugehörige Datenträgervolumen im Status **Deleting**, entfernt es aber nicht vollständig. Die Volume Group Snapshot wird entfernt, wenn alle zugehörigen Snapshots gelöscht sind.

Kubernetes `VolumeGroupSnapshotContent`-Objekte

Ein Kubernetes `VolumeGroupSnapshotContent`-Objekt repräsentiert einen Gruppen-Snapshot, der von einem bereits bereitgestellten Volume erstellt wurde. Es ist analog zu einem `PersistentVolume` und kennzeichnet einen bereitgestellten Snapshot im Speichercluster. Ähnlich wie `PersistentVolumeClaim` und `PersistentVolume` Objekten, wenn ein Snapshot erstellt wird, verwaltet das `VolumeSnapshotContent` Objekt eine Eins-zu-Eins-Zuordnung zu dem `VolumeSnapshot` Objekt, das die Snapshot-Erstellung angefordert hat.

Das `VolumeGroupSnapshotContent` Objekt enthält Details, die die Snapshot-Gruppe identifizieren, wie zum Beispiel das `volumeGroupSnapshotHandle` und einzelne volumeSnapshotHandles, die auf dem Speichersystem vorhanden sind.

Wenn eine Snapshot-Anfrage eingeht, erstellt Trident den Volume-Group-Snapshot im Backend. Nachdem der Volume-Group-Snapshot erstellt wurde, konfiguriert Trident ein `VolumeGroupSnapshotContent` Objekt und stellt den Snapshot somit der Kubernetes-API zur Verfügung.

Kubernetes `CustomResourceDefinition`-Objekte

Kubernetes Custom Resources sind Endpunkte in der Kubernetes-API, die vom Administrator definiert werden und zur Gruppierung ähnlicher Objekte verwendet werden. Kubernetes unterstützt die Erstellung von Custom Resources zum Speichern einer Sammlung von Objekten. Sie können diese Ressourcendefinitionen abrufen, indem Sie `kubectl get crds` ausführen.

Benutzerdefinierte Ressourcendefinitionen (CRDs) und die zugehörigen Objektmetadaten werden von Kubernetes in seinem Metadatenspeicher abgelegt. Dadurch entfällt die Notwendigkeit eines separaten Speichers für Trident.

Trident verwendet `CustomResourceDefinition` Objekte, um die Identität von Trident-Objekten wie Trident Backends, Trident Storage Classes und Trident Volumes zu erhalten. Diese Objekte werden von Trident verwaltet. Darüber hinaus führt das CSI Volume Snapshot Framework einige CRDs ein, die zur Definition von Volume Snapshots erforderlich sind.

CRDs sind ein Konstrukt von Kubernetes. Objekte der oben definierten Ressourcen werden von Trident erstellt. Als einfaches Beispiel: Wenn ein Backend mit `tridentctl` erstellt wird, wird ein entsprechendes `tridentbackends` CRD-Objekt zur Verwendung durch Kubernetes erzeugt.

Hier sind einige Punkte, die Sie bei den CRDs von Trident beachten sollten:

- Bei der Installation von Trident wird ein Satz von CRDs erstellt, die wie jeder andere Ressourcentyp verwendet werden können.
- Bei der Deinstallation von Trident mit dem `tridentctl uninstall`-Befehl werden die Trident-Pods gelöscht, aber die erstellten CRDs werden nicht bereinigt. Siehe "Trident deinstallieren", um zu verstehen, wie Trident vollständig entfernt und von Grund auf neu konfiguriert werden kann.

Trident `StorageClass` Objekte

Trident erstellt passende Speicherklassen für Kubernetes `StorageClass`-Objekte, die `csi.trident.netapp.io` in ihrem Provisioner-Feld angegeben sind. Der Name der Speicherklasse entspricht dem Namen des Kubernetes `StorageClass`-Objekts, das sie repräsentiert.

HINWEIS	Mit Kubernetes werden diese Objekte automatisch erstellt, wenn ein Kubernetes `StorageClass`, das Trident als Provisioner verwendet, registriert wird.

Speicherklassen umfassen eine Reihe von Anforderungen für Volumes. Trident gleicht diese Anforderungen mit den Attributen ab, die in jedem Speicherpool vorhanden sind; wenn sie übereinstimmen, ist dieser Speicherpool ein gültiges Ziel für die Bereitstellung von Volumes mit dieser Speicherklasse.

Sie können Speicherklassenkonfigurationen erstellen, um Speicherklassen direkt über die REST-API zu definieren. Bei Kubernetes-Bereitstellungen erwarten wir jedoch, dass sie bei der Registrierung neuer Kubernetes `StorageClass` Objekte erstellt werden.

Trident Backend-Objekte

Backends stellen die Speicheranbieter dar, auf denen Trident Volumes bereitstellt; eine einzelne Trident Instanz kann beliebig viele Backends verwalten.

HINWEIS	Dies ist einer der beiden Objekttypen, die Sie selbst erstellen und verwalten. Der andere ist das Kubernetes `StorageClass`-Objekt.

Weitere Informationen darüber, wie Sie diese Objekte erstellen, finden Sie unter "Backends konfigurieren".

Trident `StoragePool` Objekte

Speicherpools repräsentieren die verschiedenen, für die Bereitstellung auf jedem Backend verfügbaren Speicherorte. Für ONTAP entsprechen diese Aggregaten in SVMs. Für NetApp HCI/SolidFire entsprechen sie den vom Administrator festgelegten QoS-Bändern. Jeder Speicherpool verfügt über eine Reihe spezifischer Speicherattribute, die seine Leistungs- und Datenschutzmerkmale definieren.

Im Gegensatz zu den anderen Objekten hier werden Speicherpool-Kandidaten immer automatisch erkannt und verwaltet.

Trident `Volume` Objekte

Volumes sind die grundlegende Bereitstellungseinheit und umfassen Backend-Endpunkte wie NFS-Freigaben sowie iSCSI- und FC-LUNs. In Kubernetes entsprechen diese direkt `PersistentVolumes`. Wenn Sie ein Volume erstellen, stellen Sie sicher, dass es eine Speicherklasse hat, die bestimmt, wo dieses Volume bereitgestellt werden kann, sowie eine Größe.

HINWEIS	• In Kubernetes werden diese Objekte automatisch verwaltet. Sie können sie anzeigen, um zu sehen, was Trident bereitgestellt hat. • Beim Löschen eines Persistent Volume (PV) mit zugehörigen Snapshots wird das entsprechende Trident-Volume auf den Status **Deleting** aktualisiert. Damit das Trident-Volume gelöscht werden kann, sollten Sie die Snapshots des Volumes entfernen.

Eine Volumenkonfiguration definiert die Eigenschaften, die ein bereitgestelltes Volumen haben sollte.

Attribut	Typ	Erforderlich	Beschreibung
Version	Zeichenkette	nein	Version der Trident API ("1")
Name	Zeichenkette	Ja	Name des zu erstellenden Volumes
storageClass	Zeichenkette	Ja	Speicherklasse, die beim Bereitstellen des Volumes verwendet werden soll
Größe	Zeichenkette	Ja	Größe des Volumens, das in Bytes bereitgestellt werden soll
Protokoll	Zeichenkette	nein	Zu verwendender Protokolltyp; „Datei“ oder „Block“
internalName	Zeichenkette	nein	Name des Objekts auf dem Speichersystem; generiert von Trident

Attribut	Typ	Erforderlich	Beschreibung
cloneSourceVolume	Zeichenkette	nein	ontap (nas, san) & solidfire-*: Name des Volumes, von dem geklont werden soll
splitOnClone	Zeichenkette	nein	ontap (nas, san): Trennen Sie den Klon von seinem übergeordneten Objekt
snapshotPolicy	Zeichenkette	nein	ontap-*: Zu verwendende Snapshot-Richtlinie
snapshotReserve	Zeichenkette	nein	ontap-*: Prozentsatz des für Snapshots reservierten Volumens
exportPolicy	Zeichenkette	nein	ontap-nas*: Zu verwendende Exportrichtlinie
snapshotDirectory	bool	nein	ontap-nas*: Ob das Snapshot-Verzeichnis sichtbar ist
unixPermissions	Zeichenkette	nein	ontap-nas*: Initial UNIX-Berechtigungen
blockSize	Zeichenkette	nein	solidfire-*: Block-/Sektorgröße
fileSystem	Zeichenkette	nein	Dateisystemtyp
skipRecoveryQueue	Zeichenkette	nein	Während des Löschens eines Volumes wird die Wiederherstellungswartes chlange im Speicher umgangen und das Volume sofort gelöscht.

Trident generiert `internalName` beim Erstellen des Volumes einen Namen. Dies besteht aus zwei Schritten. Zuerst wird dem Volume-Namen das Speicherpräfix vorangestellt (entweder das Standard `trident` oder das in der Backend-Konfiguration festgelegte Präfix), sodass ein Name der Form `<prefix>-<volume-name>` entsteht. Anschließend wird der Name bereinigt, indem Zeichen ersetzt werden, die im Backend nicht zulässig sind. Für ONTAP-Backends werden Bindestriche durch Unterstriche ersetzt (dadurch wird der interne Name zu `<prefix>_<volume-name>`). Für Element-Backends werden Unterstriche durch Bindestriche ersetzt.

Sie können Volume-Konfigurationen verwenden, um Volumes direkt über die REST-API bereitzustellen, aber bei Kubernetes-Bereitstellungen gehen wir davon aus, dass die meisten Benutzer die Standardmethode von Kubernetes `PersistentVolumeClaim` verwenden. 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 zur Wiederherstellung des Zustands verwendet werden können. In Kubernetes entsprechen diese direkt

`VolumeSnapshotContent` Objekten. Jeder Snapshot ist einem Volume zugeordnet, das die Quelle der Daten für den Snapshot ist.

Jedes `Snapshot` Objekt umfasst die unten aufgeführten Eigenschaften:

Attribut	Typ	Erforderlich	Beschreibung
Version	Zeichenkette	Ja	Version der Trident API ("1")
Name	Zeichenkette	Ja	Name des Trident Snapshot-Objekts
internalName	Zeichenkette	Ja	Name des Trident-Snapshot-Objekts auf dem Speichersystem
volumeName	Zeichenkette	Ja	Name des Persistent Volume, für das der Snapshot erstellt wird
volumeInternalName	Zeichenkette	Ja	Name des zugehörigen Trident volume object im Speichersystem

HINWEIS In Kubernetes werden diese Objekte automatisch verwaltet. Sie können sie anzeigen, um zu sehen, was Trident bereitgestellt hat.

Wenn eine Kubernetes `VolumeSnapshot`-Objektanforderung erstellt wird, arbeitet Trident, indem es ein Snapshot-Objekt auf dem zugrunde liegenden Speichersystem erstellt. Die `internalName` dieses Snapshot-Objekts wird durch die Kombination des Präfixes `snapshot-` mit der `UID` des `VolumeSnapshot` Objekts generiert (zum Beispiel, `snapshot-e8d8a0ca-9826-11e9-9807-525400f3f660`). `volumeName` und `volumeInternalName` werden durch das Abrufen der Details des zugrunde liegenden Volumes befüllt.

Trident `ResourceQuota` Objekt

Der Trident Daemonset verwendet eine `system-node-critical` Priority Class – die höchste in Kubernetes verfügbare Priority Class –, um sicherzustellen, dass Trident Volumes während des ordnungsgemäßen Herunterfahrens von Knoten identifizieren und bereinigen kann und damit Trident Daemonset Pods Workloads mit niedrigerer Priorität in Clustern mit hohem Ressourcendruck verdrängen können.

Um dies zu erreichen, verwendet Trident ein `ResourceQuota` Objekt, um sicherzustellen, dass die „system-node-critical“ Priority Class beim Trident-Daemonset erfüllt ist. Vor der Bereitstellung und der Erstellung des Daemonsets sucht Trident nach dem `ResourceQuota` Objekt und wendet es an, falls es nicht gefunden wird.

Wenn Sie mehr Kontrolle über das Standardressourcenkontingent und die Prioritätsklasse benötigen, können Sie eine `custom.yaml` oder das `ResourceQuota` Objekt mithilfe eines Helm-Charts generieren oder konfigurieren.

Nachfolgend ein Beispiel für ein `ResourceQuota`-Objekt, das dem Trident daemonset Priorität einräumt.

```
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 Ressourcenquoten finden Sie unter "Kubernetes: Ressourcenkontingente".

Aufräumarbeiten durchführen `ResourceQuota` falls die Installation fehlschlägt

Sollte die Installation in dem seltenen Fall fehlschlagen, nachdem das `ResourceQuota` Objekt erstellt wurde, versuchen Sie zuerst "Deinstallation" und installieren Sie dann erneut.

Sollte das nicht funktionieren, entfernen Sie das `ResourceQuota` Objekt manuell.

Entfernen `ResourceQuota`

Wenn Sie Ihre Ressourcenzuweisung lieber selbst steuern möchten, können Sie das Trident `ResourceQuota` Objekt mit dem folgenden Befehl entfernen:

```
kubectl delete quota trident-csi -n trident
```

Pod-Sicherheitsstandards (PSS) und Security Context Constraints (SCC)

Die Kubernetes Pod Security Standards (PSS) und Pod Security Policies (PSP) definieren Berechtigungsstufen und schränken das Verhalten von Pods ein. OpenShift Security Context Constraints (SCC) definieren analog dazu pod-spezifische Einschränkungen für die OpenShift Kubernetes Engine. Um diese Anpassung zu ermöglichen, aktiviert Trident während der Installation bestimmte Berechtigungen. Die folgenden Abschnitte beschreiben die von Trident gesetzten Berechtigungen.

HINWEIS	PSS ersetzt Pod Security Policies (PSP). PSP wurde in Kubernetes v1.21 als veraltet markiert und wird in v1.25 entfernt. Weitere Informationen finden Sie unter "Kubernetes: Sicherheit".

Erforderlicher Kubernetes-Sicherheitskontext und zugehörige Felder

Berechtigung	Beschreibung
Privilegiert	CSI erfordert bidirektionale Mountpunkte, was bedeutet, dass der Trident-Node-Pod einen privilegierten Container ausführen muss. Weitere Informationen finden sich unter "Kubernetes: Mount Propagation".
Host-Netzwerk	Erforderlich für den iSCSI-Daemon. `iscsiadm` Verwaltet iSCSI-Mounts und nutzt das Host-Netzwerk zur Kommunikation mit dem iSCSI-Daemon.
Host-IPC	NFS nutzt Interprozesskommunikation (IPC), um mit dem NFSD zu kommunizieren.
Host-PID	Erforderlich, um `rpc-statd` für NFS zu starten. Trident fragt Hostprozesse ab, um festzustellen, ob `rpc-statd` läuft, bevor NFS-Volumes eingebunden werden.
Fähigkeiten	Die `SYS_ADMIN` Fähigkeit wird als Teil der Standardfähigkeiten für privilegierte Container bereitgestellt. Zum Beispiel setzt Docker diese Fähigkeiten für privilegierte Container: `CapPrm: 0000003fffffffff` `CapEff: 0000003fffffffff`
Seccomp	Das Seccomp-Profil ist in privilegierten Containern immer "Unconfined"; daher kann es in Trident nicht aktiviert werden.
SELinux	Auf OpenShift werden privilegierte Container in der `spc_t` („Super Privileged Container“) Domäne ausgeführt, und unprivilegierte Container werden in der `container_t` Domäne ausgeführt. Auf `containerd`, mit `container-selinux` installiert, werden alle Container in der `spc_t` Domäne ausgeführt, was SELinux effektiv deaktiviert. Daher fügt Trident `seLinuxOptions` nicht zu Containern hinzu.
DAC	Privilegierte Container müssen als Root ausgeführt werden. Nicht privilegierte Container werden als Root ausgeführt, um auf die von CSI benötigten Unix-Sockets zuzugreifen.

Pod Security Standards (PSS)

Etikett	Beschreibung	Standard
`pod-security.kubernetes.io/enforce pod-security.kubernetes.io/enforce-version`	Ermöglicht dem Trident Controller und den Knoten die Aufnahme in den Installations-Namespace. Ändern Sie das Namespace-Label nicht.	`enforce: privileged enforce-version: <version of the current cluster or highest version of PSS tested.>`

WARNUNG Das Ändern der Namespace-Labels kann dazu führen, dass Pods nicht eingeplant werden, eine „Error creating: …“- oder „Warning: trident-csi-…“-Meldung erscheint. Wenn dies geschieht, überprüfen Sie, ob das Namespace-Label für `privileged` geändert wurde. Falls ja, installieren Sie Trident neu.

Pod-Sicherheitsrichtlinien (PSP)

Feld	Beschreibung	Standard
`allowPrivilegeEscalation`	Privilegierte Container müssen eine Privilegien-Eskalation zulassen.	`true`
`allowedCSIDrivers`	Trident verwendet keine Inline-CSI-Ephemeral-Volumes.	Leer
`allowedCapabilities`	Nicht privilegierte Trident-Container benötigen nicht mehr Berechtigungen als die Standardmenge, und privilegierten Containern werden alle möglichen Berechtigungen gewährt.	Leer
`allowedFlexVolumes`	Trident verwendet keine "FlexVolume-Treiber", daher sind sie nicht in der Liste der erlaubten Volumes.	Leer
`allowedHostPaths`	Der Trident-Node-Pod mountet das Root-Dateisystem des Nodes, daher bringt das Festlegen dieser Liste keinen Vorteil.	Leer
`allowedProcMountTypes`	Trident verwendet kein `ProcMountTypes`.	Leer
`allowedUnsafeSysctls`	Trident benötigt keine unsicheren `sysctls`.	Leer
`defaultAddCapabilities`	Für privilegierte Container müssen keine Funktionen hinzugefügt werden.	Leer
`defaultAllowPrivilegeEscalation`	Die Gewährung von Privilegienerweiterungen wird in jedem Trident Pod gehandhabt.	`false`
`forbiddenSysctls`	Keine `sysctls` sind erlaubt.	Leer

Feld	Beschreibung	Standard
`fsGroup`	Trident Container werden als root ausgeführt.	`RunAsAny`
`hostIPC`	Das Einbinden von NFS-Volumes erfordert Host-IPC, um mit `nfsd`	`true`
`hostNetwork`	iscsiadm benötigt das Host-Netzwerk, um mit dem iSCSI-Daemon zu kommunizieren.	`true`
`hostPID`	Host-PID ist erforderlich, um zu überprüfen, ob `rpc-statd` auf dem Knoten läuft.	`true`
`hostPorts`	Trident verwendet keine Host-Ports.	Leer
`privileged`	Trident node pods müssen einen privilegierten Container ausführen, um Volumes einzubinden.	`true`
`readOnlyRootFilesystem`	Trident node pods müssen in das Node-Dateisystem schreiben.	`false`
`requiredDropCapabilities`	Trident node pods führen einen privilegierten Container aus und können keine Capabilities ablegen.	`none`
`runAsGroup`	Trident Container werden als root ausgeführt.	`RunAsAny`
`runAsUser`	Trident Container werden als root ausgeführt.	`runAsAny`
`runtimeClass`	Trident verwendet `RuntimeClasses` nicht.	Leer
`seLinux`	Trident setzt `seLinuxOptions` nicht, da es derzeit Unterschiede gibt, wie Container-Runtimes und Kubernetes-Distributionen SELinux handhaben.	Leer
`supplementalGroups`	Trident Container werden als root ausgeführt.	`RunAsAny`
`volumes`	Trident Pods benötigen diese Volume-Plugins.	`hostPath, projected, emptyDir`

Security Context Constraints (SCC)

Etiketten	Beschreibung	Standard
`allowHostDirVolumePlugin`	Trident-Knoten-Pods mounten das Root-Dateisystem des Knotens.	`true`

Etiketten	Beschreibung	Standard
`allowHostIPC`	Das Mounten von NFS-Volumes erfordert Host-IPC, um mit `nfsd` zu kommunizieren.	`true`
`allowHostNetwork`	iscsiadm benötigt das Host-Netzwerk, um mit dem iSCSI-Daemon zu kommunizieren.	`true`
`allowHostPID`	Host-PID ist erforderlich, um zu überprüfen, ob `rpc-statd` auf dem Knoten läuft.	`true`
`allowHostPorts`	Trident verwendet keine Host-Ports.	`false`
`allowPrivilegeEscalation`	Privilegierte Container müssen eine Privilegien-Eskalation zulassen.	`true`
`allowPrivilegedContainer`	Trident node pods müssen einen privilegierten Container ausführen, um Volumes einzubinden.	`true`
`allowedUnsafeSysctls`	Trident benötigt keine unsicheren `sysctls`.	`none`
`allowedCapabilities`	Nicht privilegierte Trident-Container benötigen nicht mehr Berechtigungen als die Standardmenge, und privilegierten Containern werden alle möglichen Berechtigungen gewährt.	Leer
`defaultAddCapabilities`	Für privilegierte Container müssen keine Funktionen hinzugefügt werden.	Leer
`fsGroup`	Trident Container werden als root ausgeführt.	`RunAsAny`
`groups`	Diese SCC ist spezifisch für Trident und ist an ihren Benutzer gebunden.	Leer
`readOnlyRootFilesystem`	Trident node pods müssen in das Node-Dateisystem schreiben.	`false`
`requiredDropCapabilities`	Trident node pods führen einen privilegierten Container aus und können keine Capabilities ablegen.	`none`
`runAsUser`	Trident Container werden als root ausgeführt.	`RunAsAny`
`seLinuxContext`	Trident setzt `seLinuxOptions` nicht, da es derzeit Unterschiede gibt, wie Container-Runtimes und Kubernetes-Distributionen SELinux handhaben.	Leer

Etiketten	Beschreibung	Standard
`seccompProfiles`	Privilegierte Container laufen immer "Unconfined".	Leer
`supplementalGroups`	Trident Container werden als root ausgeführt.	`RunAsAny`
`users`	Ein Eintrag wird bereitgestellt, um diese SCC an den Trident Benutzer im Trident-Namensraum zu binden.	n/a
`volumes`	Trident Pods benötigen diese Volume-Plugins.	`hostPath, downwardAPI, projected, emptyDir`

Rechtliche Hinweise

Rechtliche Hinweise bieten Zugriff auf Urheberrechtsvermerke, Marken, Patente und mehr.

Copyright

"https://www.netapp.com/company/legal/copyright/"

Marken

NETAPP, das NETAPP-Logo und die auf der NetApp Trademarks-Seite aufgeführten Marken sind Marken von NetApp, Inc. Andere Firmen- und Produktnamen können Marken ihrer jeweiligen Eigentümer sein.

"https://www.netapp.com/company/legal/trademarks/"

Patente

Eine aktuelle Liste der NetApp owned patents 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 die Urheberrechte und Lizenzen Dritter, die in NetApp-Software für Trident verwendet werden, in der Mitteilungsdatei für jede Version unter https://github.com/NetApp/trident/ einsehen.

Copyright-Informationen

Copyright © 2026 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.